RWTH Aachen

Department of Computer Science
Technical Report

Size-Change Termination for Term Rewriting

René Thiemann and Jürgen Giesl

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2003-2

RWTH Aachen · Department of Computer Science · April 2003 (revised version)

Size-Change Termination for Term Rewriting*

René Thiemann and Jürgen Giesl

LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
{thiemann|giesl}@informatik.rwth-aachen.de

Abstract. In [13], a new *size-change principle* was proposed to verify termination of functional programs automatically. We extend this principle in order to prove termination and innermost termination of arbitrary term rewrite systems (TRSs). Moreover, we compare this approach with existing techniques for termination analysis of TRSs (such as recursive path orderings or dependency pairs). It turns out that the size-change principle on its own fails for many examples that can be handled by standard techniques for rewriting, but there are also TRSs where it succeeds whereas existing rewriting techniques fail. In order to benefit from their respective advantages, we show how to combine the size-change principle with classical orderings and with dependency pairs. In this way, we obtain a new approach for automated termination proofs of TRSs which is more powerful than previous approaches.

1 Introduction

The size-change principle [13] is a new technique for automated termination analysis of functional programs, which raised great interest in the functional programming and automated reasoning community. However, up to now the connection between this principle and existing approaches for termination proofs of term rewriting was unclear. After introducing the size-change principle in Sect. 2, we show how to use it for (innermost) termination proofs of arbitrary TRSs in Sect. 3. This also illustrates how to combine the size-change principle with existing orderings from term rewriting. In Sect. 4 and 5 we compare the size-change principle with classical simplification orderings and with the dependency pair approach [1] for termination of TRSs. Finally, to combine their advantages, we developed a technique which integrates the size-change principle and dependency pairs. The combined technique has been implemented in the system AProVE resulting in a very efficient and powerful automated method which improves the original dependency pair approach significantly. A description of the implementation can be found in the appendix.

2 The Size-Change Principle

We assume familiarity with the basics of term rewriting (see e.g., [3]). For a TRS $\mathcal{R}$ over a signature $\mathcal{F}$, the *defined symbols* $\mathcal{D}$ are the root symbols of the left-hand sides of rules and the *constructors* are $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$. We restrict ourselves to

* Extended version of a paper from the *Proceedings of the 14th Int. Conference on Rewriting Techniques and Applications (RTA-03)*, Valencia, Spain, LNCS, Springer-Verlag, 2003.

finite signatures and TRSs. A TRS is called a *constructor system* if the left-hand sides of its rules are terms of the form $f(s_1, \ldots, s_n)$ where all s_i are *constructor terms* (i.e., $s_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$). For any signature $\mathcal{F}$ we define the *embedding rules* $Emb_{\mathcal{F}} = \{f(x_1, \ldots, x_n) \to x_i \mid f \in \mathcal{F}$ where $n = \text{arity}(f), 1 \leq i \leq n\}$.

In [13], the size-change principle was formulated for a functional programming language with eager evaluation strategy and without pattern matching. Such functional programs are easily transformed into TRSs which are orthogonal constructor systems whose ground normal forms only contain constructors (i.e., all functions are "completely" defined). In this section we present an extension of the original size-change principle which can be used for arbitrary TRSs.

We call $(\succsim, \succ)$ a *reduction pair* [11] on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ if $\succsim$ is a quasi-ordering and $\succ$ is a well-founded ordering on terms where both $\succsim$ and $\succ$ are closed under substitutions and compatible (i.e., $\succsim \circ \succ \subseteq \succ$ or $\succ \circ \succsim \subseteq \succ$, but $\succ \subseteq \succsim$ is not required). In general, neither $\succsim$ nor $\succ$ have to be closed under contexts. If $\succsim$ is closed under contexts, we speak of a *monotonic* reduction pair. In Sect. 3 we examine which additional conditions must be imposed on $(\succsim, \succ)$ in order to use the size-change principle for (innermost) termination proofs of TRSs. *Size-change graphs* denote how the size of function parameters changes when going from one function call to another.

Definition 1 (Size-Change Graph) *Let $(\succsim, \succ)$ be a reduction pair. For every rule $f(s_1, \ldots, s_n) \to r$ of a TRS $\mathcal{R}$ and every subterm $g(t_1, \ldots, t_m)$ of r where $g \in \mathcal{D}$, we define a* size-change graph. *The graph has n* output nodes *marked with $\{1_f, \ldots, n_f\}$ and m* input nodes *marked with $\{1_g, \ldots, m_g\}$. If $s_i \succ t_j$, then there is a directed edge marked with "$\succ$" from output node i_f to input node j_g. Otherwise, if $s_i \succsim t_j$, then there is an edge marked with "$\succsim$" from i_f to j_g. If f and g are clear from the context, then we often omit the subscripts from the nodes. So a size-change graph is a bipartite graph $G = (V, W, E)$ where $V = \{1_f, \ldots, n_f\}$ and $W = \{1_g, \ldots, m_g\}$ are the labels of the output and input nodes, respectively, and we have edges $E \subseteq V \times W \times \{\succsim, \succ\}$.*

Example 2 Let $\mathcal{R}$ consist of the following rules.

$$\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(x, \mathsf{s}(x)) \qquad (1) \qquad\qquad \mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(y, x) \qquad (2)$$

$\mathcal{R}$ has two size-change graphs $G_{(1)}$ and $G_{(2)}$ resulting from (1) and (2). Here, we use the embedding ordering on constructors $\mathcal{C}$, i.e., $(\succsim, \succ) = (\to^*_{Emb_{\mathcal{C}}}, \to^+_{Emb_{\mathcal{C}}})$.

To trace sizes of parameters along subsequent function calls, size-change graphs (V_1, W_1, E_1) and (V_2, W_2, E_2) can be concatenated to *multigraphs* if $W_1 = V_2$, i.e., if they correspond to arguments $\{1_g, \ldots, m_g\}$ of the same function g.

Definition 3 (Multigraph and Concatenation) *Every size-change graph of $\mathcal{R}$ is a* multigraph *of $\mathcal{R}$ and if $G = (\{1_f, \ldots, n_f\}, \{1_g, \ldots, m_g\}, E_1)$ and $H =$*

$(\{1_g, \ldots, m_g\}, \{1_h, \ldots, p_h\}, E_2)$ *are multigraphs w.r.t. the same reduction pair* $(\succsim, \succ)$*, then the* concatenation $G \bullet H = (\{1_f, \ldots, n_f\}, \{1_h, \ldots, p_h\}, E)$ *is also a multigraph of* $\mathcal{R}$*. For* $1 \leq i \leq n$ *and* $1 \leq k \leq p$*,* E *contains an edge from* i_f *to* k_h *iff* E_1 *contains an edge from* i_f *to some* j_g *and* E_2 *contains an edge from* j_g *to* k_h*. If there is such a* j_g *where the edge of* E_1 *or* E_2 *is labeled with "*$\succ$*", then the edge in* E *is labeled with "*$\succ$*" as well. Otherwise, it is labeled with "*$\succsim$*". A multigraph* G *is called* maximal *if its input and output nodes are both labeled with* $\{1_f, \ldots, n_f\}$ *for some* f *and if it is idempotent, i.e.,* $G = G \bullet G$*.*

Example 4 In Ex. 2 we obtain the following three maximal multigraphs:

$$G_{(1)} \bullet G_{(2)} : \begin{tikzcd} 1_\mathsf{f} \arrow[r, "\succ"] \arrow[dr, "\succ"] & 1_\mathsf{f} \\ 2_\mathsf{f} & 2_\mathsf{f} \end{tikzcd} \qquad G_{(2)} \bullet G_{(1)} : \begin{tikzcd} 1_\mathsf{f} & 1_\mathsf{f} \\ 2_\mathsf{f} \arrow[ur, "\succ"] \arrow[r, "\succ"'] & 2_\mathsf{f} \end{tikzcd} \qquad G_{(2)} \bullet G_{(2)} : \begin{tikzcd} 1_\mathsf{f} \arrow[r, "\succ"] & 1_\mathsf{f} \\ 2_\mathsf{f} \arrow[r, "\succ"] & 2_\mathsf{f} \end{tikzcd}$$

For termination, in every maximal multigraph a parameter must be decreasing.

Definition 5 (Size-Change Termination) *A TRS* $\mathcal{R}$ *over the signature* $\mathcal{F}$ *is* size-change terminating *w.r.t. a reduction pair* $(\succsim, \succ)$ *on* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *iff every maximal multigraph contains an edge of the form* $i \xrightarrow{\succ} i$*.*

In Ex. 4, each maximal multigraph contains the edge $1_\mathsf{f} \xrightarrow{\succ} 1_\mathsf{f}$ or $2_\mathsf{f} \xrightarrow{\succ} 2_\mathsf{f}$. So the TRS is size-change terminating w.r.t. the embedding ordering. Note that classical path orderings from term rewriting fail on this example (see Sect. 4).

Since there are only finitely many possible multigraphs, they can be constructed automatically. So for a given reduction pair, size-change termination is decidable. However, in general size-change termination does not imply termination.

Example 6 Consider the TRS with the rules $\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b})$ and $\mathsf{b} \to \mathsf{a}$. If we use the lexicographic path ordering $\succ_{LPO}$ [9] with the precedence $\mathsf{a} > \mathsf{b}$, then the only maximal multigraph is $1_\mathsf{f} \xrightarrow{\succ_{LPO}} 1_\mathsf{f}$. So size-change termination is proved, although the TRS is obviously not terminating.

In [13], size-change termination was defined in a slightly different way. Here, instead of concatenating size-change graphs $G_1, \ldots, G_n$, one builds (possibly infinite) graphs by identifying the input nodes of a size-change graph with the output nodes of the next size-change graph. They called a program *size-change terminating* if there exists an infinite path in this graph which contains infinitely many edges labeled with "$\succ$". The following lemma (which corresponds to [13, Thm. 4]) states that our definition is equivalent to the one of [13].

Lemma 7 (Correspondence of Infinite Graphs and Multigraphs [13]) *Let* Γ *be a finite set of size-change graphs. Then the following two statements are equivalent.*

(1) In every infinite graph resulting from graphs of Γ *by identifying the input nodes of a graph with the output nodes of the next graph, there exists an infinite path containing infinitely many edges labeled with "*$\succ$*".*

(2) Every maximal multigraph resulting from concatenations of graphs of Γ has an edge of the form $i \overset{\succsim}{\rightarrow} i$.

Proof. (1) $\Rightarrow$ (2): For two size-change graphs or multigraphs G and H where G's input nodes have the same labels as H's output nodes, let $G \circ H$ be the graph resulting from identifying G's input and H's output nodes. So $G \circ H$ differs from $G \bullet H$ in that these nodes are not dropped.

Assume that there exists a maximal multigraph $G = G_1 \bullet \ldots \bullet G_n$ which has no edge of the form $i \overset{\succ}{\rightarrow} i$. On the other hand, the infinite graph $G_1 \circ \ldots \circ G_n \circ G_1 \circ \ldots \circ G_n \circ \ldots$ must have infinitely many edges labeled with "$\succ$". Thus, this also holds for the infinite graph $G \circ G \circ \ldots$ Obviously, for some $i \in \mathbb{N}$ and $f \in \mathcal{F}$, a node labeled with i_f must occur more than once in this path such that an edge between these two occurrences is labeled with "$\succ$". Let n be the length of the subpath from the first occurrence of i_f to the next occurrence of i_f such that an $\overset{\succ}{\rightarrow}$-edge is on this subpath. Thus, there is a path from i_f to i_f in the graph $G \circ G \circ \ldots \circ G$ (where G is combined with itself n times) and at least one edge of the path is labeled with "$\succ$". This means that the multigraph $G \bullet G \bullet \ldots \bullet G$ (where G is concatenated with itself n times), contains an edge $i \overset{\succ}{\rightarrow} i$. Since G is idempotent, we have $G \bullet G \bullet \ldots \bullet G = G$ and thus, this contradicts the assumption that G does not have such edges.

(2) $\Rightarrow$ (1): Assume that there is an infinite graph $G_1 \circ G_2 \circ \ldots$ that does not contain an infinite path with infinitely many "$\succ$" edges. For all pairs of numbers (n, m) with $n < m$ let $G_{n,m}$ be the multigraph resulting from the concatenation of $G_n, \ldots, G_{m-1}$, i.e., $G_{n,m} = G_n \bullet \ldots \bullet G_{m-1}$. As there are only finitely many possible multigraphs, by Ramsey's theorem there is an infinite set $I \subseteq \mathbb{N}$ such that $G_{n,m}$ is always the same graph for all $n, m \in I$ with $n < m$. We call this graph G. Note that G is a maximal multigraph: for $n_0 < n_1 < n_2$ with $n_i \in I$, we have $G_{n_0,n_2} = G_{n_0} \bullet \ldots \bullet G_{n_1-1} \bullet G_{n_1} \bullet \ldots \bullet G_{n_2-1} = G_{n_0,n_1} \bullet G_{n_1,n_2}$, and thus $G = G \bullet G$.

Let $I = \{n_0, n_1, \ldots\}$. Thus, for our original infinite graph, we have

$$G_1 \circ G_2 \circ \ldots \;=\; G_1 \circ \ldots \circ G_{n_0-1} \circ G_{n_0} \circ \ldots \circ G_{n_1-1} \circ G_{n_1} \circ \ldots \circ G_{n_2-1} \circ \ldots$$

Since by assumption, this graph did not contain an infinite path with infinitely many "$\succ$" edges, this also holds for the graph

$$G_{n_0} \bullet \ldots \bullet G_{n_1-1} \circ G_{n_1} \bullet \ldots \bullet G_{n_2-1} \circ \ldots \;=\; G_{n_0,n_1} \circ G_{n_1,n_2} \circ \ldots \;=\; G \circ G \circ \ldots$$

But since G is a maximal multigraph, G contains an edge $i \overset{\succ}{\rightarrow} i$. Thus, in contradiction to the assumption, the above infinite graph does contain an infinite graph labeled with infinitely many "$\succ$" edges. $\square$

3 Size-Change Termination and Termination of TRSs

In this section we develop conditions on the reduction pair used in Def. 5 which ensure that size-change termination indeed implies (innermost) termination.

Then the size-change principle can be combined with classical orderings from term rewriting and it becomes a sound termination criterion.

In [13], the authors use reduction pairs $(\succsim, \succ)$ where $\succsim$ is the reflexive closure of $\succ$ and $\succ$ is defined in terms of a well-founded relation $>$ on (ground) normal forms of $\mathcal{R}$. We now show that such reduction pairs can be used for innermost termination proofs of arbitrary TRSs. For non-overlapping systems as in [13], it suffices to regard *ground* normal forms, since there, ground innermost termination is equivalent to innermost termination (and in fact, to termination). For arbitrary TRSs, one has to regard normal forms with variables as well. Moreover, $\succsim$ can be any compatible quasi-ordering. We denote innermost reduction steps by $\xrightarrow{\text{i}}_{\mathcal{R}}$ and $s \xrightarrow{\text{i}}{}^!_{\mathcal{R}} s'$ means that s' is a normal form reachable from s by innermost reduction. Thm. 8 will serve as the basis for the automation of the size-change principle in Thm. 9 afterwards.

Theorem 8 (Size-Change Termination and Innermost Termination) *Let $>$ be a well-founded ordering on normal forms of a TRS $\mathcal{R}$. For $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ we define* $\mathrm{NF}(s,t) = \{(s', t') \mid s\sigma \xrightarrow{\text{i}}{}^!_{\mathcal{R}} s', t\sigma \xrightarrow{\text{i}}{}^!_{\mathcal{R}} t', \sigma$ *instantiates all variables of s and t with normal forms of $\mathcal{R}$*$\}$*. Let $(\succsim, \succ)$ be a reduction pair where $s \succ t$ implies $s' > t'$ for all $(s', t') \in \mathrm{NF}(s,t)$. If $\mathcal{R}$ is size-change terminating w.r.t. $(\succsim, \succ)$, then $\mathcal{R}$ is innermost terminating.*

Proof. If $\mathcal{R}$ is not innermost terminating, then there is a minimal non-innermost terminating term v_0, i.e., all proper subterms of v_0 are innermost terminating. Let $\xrightarrow{\text{i}}_{\epsilon}$ denote root reductions and let $\xrightarrow{\text{i}}_{>\epsilon}$ denote reductions below the root. Then v_0's infinite innermost reduction starts with $v_0 \xrightarrow{\text{i}}{}^*_{>\epsilon} u_1 \xrightarrow{\text{i}}_{\epsilon} w_1$ where all proper subterms of u_1 are in normal form. Since w_1 is not innermost terminating, it has a minimal non-innermost terminating subterm v_1.

The infinite reduction continues in the same way. So for $i \geq 1$, we have $v_{i-1} \xrightarrow{\text{i}}{}^*_{>\epsilon} u_i = l_i\sigma_i$ and $v_i = r'_i\sigma_i$ for a rule $l_i \to r_i$, a subterm r'_i of r_i with defined root, and a substitution σ_i instantiating l_i's variables with normal forms.

For each step from u_i to v_i there is a corresponding size-change graph G_i. We regard the infinite graph resulting from $G_1, G_2, \ldots$ by identifying the input nodes of G_i with the output nodes of G_{i+1}. If $\mathcal{R}$ is size-change terminating, by Lemma 7 this infinite graph contains an infinite path where infinitely many edges are labeled with "$\succ$". Without loss of generality we assume that this path already starts in G_1. For every i, let a_i be the output node in G_i which is on this path. So we have $l_i|_{a_i} \succ r'_i|_{a_{i+1}}$ for all i from an infinite set $I \subseteq \mathbb{N}$ and $l_i|_{a_i} \succsim r'_i|_{a_{i+1}}$ for $i \in \mathbb{N} \setminus I$. Note that $l_i|_{a_i}\sigma_i = u_i|_{a_i}$ and $r'_i|_{a_{i+1}}\sigma_i = v_i|_{a_{i+1}} \xrightarrow{\text{i}}{}^!_{\mathcal{R}} u_{i+1}|_{a_{i+1}}$. Thus, $(u_i|_{a_i}, u_{i+1}|_{a_{i+1}}) \in \mathrm{NF}(l_i|_{a_i}, r'_i|_{a_{i+1}})$. Hence, for $I = \{i_1, i_2, \ldots\}$ we obtain $u_{i_1}|_{a_{i_1}} > u_{i_2}|_{a_{i_2}} > \ldots$ which is a contradiction to the well-foundedness of $>$. $\square$

Innermost termination is interesting, since then there are no infinite reductions w.r.t. eager evaluation strategies. Moreover, for non-overlapping TRSs, innermost termination already implies termination. However, Thm. 8 is not yet suitable for automation. To check whether $\succ$ satisfies the conditions of Thm. 8, one has to examine infinitely many instantiations of s and t and compute normal forms s' and t' although $\mathcal{R}$ is possibly not innermost terminating. Therefore, in the examples of [13], one is restricted to relations $\succsim$ and $\succ$ on constructor terms.

Thm. 9 shows how to use such reduction pairs on $\mathcal{T}(\mathcal{C},\mathcal{V})$ for possibly automated innermost termination proofs. In general, a reduction pair $(\succsim,\succ)$ on $\mathcal{T}(\mathcal{G},\mathcal{V})$ with $\mathcal{G}\subseteq\mathcal{F}$ can be *extended* to a (usually non-monotonic) reduction pair $(\succsim',\succ')$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$ by defining $s\succsim' t$ if $s=t$ or if there exist $u,v\in\mathcal{T}(\mathcal{G},\mathcal{V})$ with $u\succsim v$ such that $s=u\sigma$ and $t=v\sigma$ for some substitution σ. Moreover, $s\succ' t$ iff $u\succ v$ for u and v as above.

Theorem 9 (Innermost Termination Proofs) *Let $(\succsim,\succ)$ be a reduction pair on $\mathcal{T}(\mathcal{C},\mathcal{V})$. If $\mathcal{R}$ is size-change terminating w.r.t. the extension of the reduction pair $(\succsim,\succ)$ to $\mathcal{T}(\mathcal{F},\mathcal{V})$, then $\mathcal{R}$ is innermost terminating.*

Proof. Let $(\succsim',\succ')$ be the extension of $(\succsim,\succ)$ to $\mathcal{T}(\mathcal{F},\mathcal{V})$. We show that $s\succ' t$ implies $s'\succ' t'$ for all $(s',t')\in\mathrm{NF}(s,t)$. Then the theorem follows from Thm. 8.

By the definition of extensions, $s\succ' t$ iff $s=u\sigma$, $t=v\sigma$, and $u\succ v$ for suitable u, v, and σ. In particular, u and v must be constructor terms and we also have $u\succ' v$ (as σ may also be the identity). Since $\mathrm{NF}(s,t)\subseteq\{(u\sigma,v\sigma)\mid$ σ instantiates u's and v's variables with normal forms$\}$, the claim follows from $u\succ' v$, because $\succ'$ is closed under substitutions. □

For the TRS in Ex. 2, when using the extension of the reduction pair $(\to^*_{Emb_\mathcal{C}},\to^+_{Emb_\mathcal{C}})$ on $\mathcal{T}(\mathcal{C},\mathcal{V})$, we obtain the same size-change graphs as with $(\to^*_{Emb_\mathcal{C}},\to^+_{Emb_\mathcal{C}})$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$. Ex. 4 shows that this TRS is size-change terminating w.r.t. this reduction pair and hence, by Thm. 9, this proves innermost termination. However, a variant of Toyama's example [14] shows that Thm. 8 and Thm. 9 are not sufficient to prove full (non-innermost) termination.

Example 10 Let $\mathcal{R}=\{\mathsf{f}(\mathsf{c}(\mathsf{a},\mathsf{b},x))\to\mathsf{f}(\mathsf{c}(x,x,x)),\ \mathsf{g}(x,y)\to x,\ \mathsf{g}(x,y)\to y\}$. We define $\succsim=\to^*_\mathcal{S}$ and $\succ=\to^+_\mathcal{S}$ restricted to $\mathcal{T}(\mathcal{C},\mathcal{V})$, where $\mathcal{S}$ is the terminating TRS with the rule $\mathsf{c}(\mathsf{a},\mathsf{b},x)\to\mathsf{c}(x,x,x)$. The only maximal multigraph is $1_\mathsf{f}\xrightarrow{\succ}1_\mathsf{f}$. Thus, $\mathcal{R}$ is size-change terminating and by Thm. 9 it is innermost terminating. However, $\mathcal{R}$ does not terminate.

As in Ex. 10, reduction pairs $(\to^*_\mathcal{S},\to^+_\mathcal{S})$ satisfying the conditions of Thm. 9 can be defined using a terminating TRS $\mathcal{S}$ over the signature $\mathcal{C}$. The following theorem shows that if $\mathcal{S}$ is non-duplicating, then we may use the relation $\to_\mathcal{S}$ also on terms with defined symbols and size-change termination even implies full termination. A TRS is *non-duplicating* if every variable occurs on the right-hand side of a rule at most as often as on the corresponding left-hand side. So size-change termination of the TRS in Ex. 2 and Ex. 4 using the reduction pair $(\to^*_{Emb_\mathcal{C}},\to^+_{Emb_\mathcal{C}})$ implies that the TRS is indeed terminating.

In order to prove the theorem, we need a preliminary lemma which states that minimal non-terminating terms w.r.t. $\mathcal{R}\cup\mathcal{S}$ cannot start with constructors of $\mathcal{R}$. Again, here $\mathcal{S}$ must be non-duplicating. Otherwise, in Ex. 10, $\mathsf{c}(\mathsf{a},\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{b}))$ is a minimal non-terminating term w.r.t. $\mathcal{R} \cup \mathcal{S}$ that starts with a constructor of $\mathcal{R}$.

Lemma 11 *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$ with constructors $\mathcal{C}$ and let $\mathcal{S}$ be a terminating non-duplicating TRS over $\mathcal{C}$. If $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F},\mathcal{V})$ are terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$ and $c \in \mathcal{C}$, then $c(t_1, \ldots, t_n)$ is also terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$.*

Proof. For any term $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$, let M_s be the multiset of the maximal subterms of s whose root is defined, i.e., $M_s = \{s|_\pi \mid root(s|_\pi) \in \mathcal{D}$ and for all π' above π we have $root(s|_{\pi'}) \in \mathcal{C}\}$. Moreover, let s' be the term that results from s by replacing all maximal subterms with defined root by the same fresh special variable $x_\mathcal{C}$. Let $\twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}}$ be the extension of $\rightarrow_{\mathcal{R}\cup\mathcal{S}}$ to multisets where $M \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}} M'$ iff $M = N \cup \{s\}$ and $M' = N \cup \{t_1, \ldots, t_n\}$ with $n \geq 0$ and with $s \rightarrow_{\mathcal{R}\cup\mathcal{S}} t_i$ for all i. We prove the following conjecture.

> Let $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$ such that all terms in M_s are terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$ and let $s \rightarrow_{\mathcal{R}\cup\mathcal{S}} t$. Then all terms in M_t are also terminating w.r.t. $\mathcal{R} \cup \mathcal{S}$. Moreover, $M_s \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}} M_t$ or both $M_s = M_t$ and $s' \rightarrow_\mathcal{S} t'$. (3)

Note that $\twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}}$ is well founded on multisets like M_s which only contain terminating terms. Termination of $\mathcal{S}$ implies that $\rightarrow_\mathcal{S}$ is also well founded and the lexicographic combination of two well-founded orderings preserves well-foundedness. Hence, (3) implies that if all terms in M_s are terminating, then s is terminating as well. So the lemma immediately follows from Conjecture (3).

To prove (3), we distinguish according to the position π where the reduction $s \rightarrow_{\mathcal{R}\cup\mathcal{S}} t$ takes place. If s has a defined symbol of $\mathcal{D}$ on or above position π, then this implies $M_s \twoheadrightarrow_{\mathcal{R}\cup\mathcal{S}} M_t$ and all terms in M_t are also terminating. Otherwise, if π is above all symbols of $\mathcal{D}$ in s, then $s \rightarrow_{\mathcal{R}\cup\mathcal{S}} t$ implies $s \rightarrow_\mathcal{S} t$ and $M_s \supseteq M_t$ (since $\mathcal{S}$ is non-duplicating). Moreover, $s \rightarrow_\mathcal{S} t$ also implies $s' \rightarrow_\mathcal{S} t'$. □

Now we can show the desired theorem.

Theorem 12 (Termination Proofs) *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$ with constructors $\mathcal{C}$ and let $\mathcal{S}$ be a terminating non-duplicating TRS over $\mathcal{C}$. If $\mathcal{R}$ is size-change terminating w.r.t. the reduction pair $(\rightarrow_\mathcal{S}^*, \rightarrow_\mathcal{S}^+)$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$, then $\mathcal{R}$ (and even $\mathcal{R} \cup \mathcal{S}$) is terminating.*

Proof. We define $\mathcal{R}' := \mathcal{R} \cup \mathcal{S}$. If $\mathcal{R}'$ is not terminating, then as in the proof of Thm. 8 we obtain an infinite sequence of minimal non-terminating terms u_i, v_i with $v_i \rightarrow_{>\epsilon,\mathcal{R}'}^* u_{i+1}$ where the step from u_i to v_i corresponds to a size-change graph of $\mathcal{R}'$. Thus, for all i there is a rule $l_i \rightarrow r_i$ in $\mathcal{R}'$ with $u_i = l_i\sigma_i$ and $v_i = r_i'\sigma_i$ for a subterm r_i' of r_i and a substitution σ_i.

By Lemma 11, the roots of u_i and v_i are defined symbols. Thus, all these size-change graphs are from $\mathcal{R}$. As in Thm. 8's proof, there are a_i with $l_i|_{a_i} \rightarrow_\mathcal{S}^+ r_i'|_{a_{i+1}}$

for all i from an infinite set $I \subseteq \mathbb{N}$ and $l_i|_{a_i} \to_{\mathcal{S}}^{*} r'_i|_{a_{i+1}}$ for $i \in \mathbb{N} \setminus I$. Since $\to_{\mathcal{S}}$ is closed under substitution we also have $u_i|_{a_i} \to_{\mathcal{S}}^{+} v_i|_{a_{i+1}}$ or $u_i|_{a_i} \to_{\mathcal{S}}^{*} v_i|_{a_{i+1}}$, respectively. Recall $v_i|_{a_{i+1}} \to_{\mathcal{R}'}^{*} u_{i+1}|_{a_{i+1}}$ and $\mathcal{S} \subseteq \mathcal{R}'$. So for $I = \{i_1, i_2, \ldots\}$ we have $u_{i_1}|_{a_{i_1}} \to_{\mathcal{R}'}^{+} u_{i_2}|_{a_{i_2}} \to_{\mathcal{R}'}^{+} \ldots$ contradicting the minimality of the terms u_i. □

With Thm. 9 and Thm. 12 we have two possibilities for automating the size-change principle. Note that even for innermost termination, Thm. 9 and Thm. 12 do not subsume each other. Ex. 10 cannot be handled by Thm. 12 and innermost termination of $\{\mathsf{g}(\mathsf{f}(\mathsf{a})) \to \mathsf{g}(\mathsf{f}(\mathsf{b})), \mathsf{f}(x) \to x\}$ cannot be proved with Thm. 9, since $\mathsf{f}(\mathsf{a}) \not\succ \mathsf{f}(\mathsf{b})$ for any extension $\succ$ of an ordering on constructor terms. On the other hand, termination is easily shown with Thm. 12 using $\mathcal{S} = \{\mathsf{a} \to \mathsf{b}\}$. In fact, a variant of Thm. 12 also holds for innermost termination if $\mathcal{S}$ is innermost terminating (and possibly duplicating). However, this variant only proves innermost termination of $\mathcal{R} \cup \mathcal{S}$ and in general, this does not imply innermost termination of $\mathcal{R}$.

So Thm. 9 and Thm. 12 are new contributions that show which reduction pairs are admissible in order to use size-change termination for termination or innermost termination proofs of TRSs. In this way, size-change termination can be turned into an automatic technique, since one can use classical techniques from termination analysis of term rewriting to generate suitable reduction pairs automatically.

4 Comparison with Orderings from Term Rewriting

Most traditional techniques for TRSs are based on so-called *simplification orderings* (like lexicographic or recursive path orderings (possibly with status) RPOS [5, 9], Knuth-Bendix orderings KBO [10], and most polynomial orderings [12]). A TRS is *simply terminating* iff it is compatible with a simplification ordering. Equivalently, a TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is simply terminating iff $\mathcal{R} \cup \mathit{Emb}_{\mathcal{F}}$ terminates. Thm. 13 shows that similar to these traditional techniques, the size-change principle can essentially only verify simple termination.

Theorem 13 (Size-Change Principle and Simple Termination)

(a) A TRS $\mathcal{R}$ over a signature $\mathcal{F}$ is size-change terminating w.r.t. a reduction pair $(\succsim, \succ)$ iff $\mathcal{R} \cup \mathit{Emb}_{\mathcal{F}}$ is size-change terminating w.r.t. $(\succsim, \succ)$.

(b) Let $\mathcal{S}$ be as in Thm. 12. If $\mathcal{S}$ is simply terminating and $\mathcal{R}$ is size-change terminating w.r.t. $(\to_{\mathcal{S}}^{*}, \to_{\mathcal{S}}^{+})$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$, then $\mathcal{R} \cup \mathcal{S}$ is simply terminating.

Proof. (a) The "if" direction is obvious. For the "only if" direction, note that $\mathit{Emb}_{\mathcal{F}}$ yields no new size-change graphs. But due to $\mathit{Emb}_{\mathcal{C}}$, all constructors are transformed into defined symbols. So from the $\mathcal{R}$-rules we obtain additional size-change graphs whose input nodes are labeled with (former) constructors (i.e., $1_c, \ldots, n_c$ for $c \in \mathcal{C}$). However, since output nodes are never labeled with constructors, this does not yield new maximal multigraphs (since there, output and input nodes are labeled by the same function). Hence, size-change termination is not affected when adding $Emb_{\mathcal{F}}$.

(b) As in (a), adding $Emb_{\mathcal{D}}$ to $\mathcal{R}$ yields no new size-change graphs and thus, $\mathcal{R} \cup Emb_{\mathcal{D}}$ is also size-change terminating w.r.t. $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$ and hence, also w.r.t. $(\to_{\mathcal{S}\cup Emb_{\mathcal{C}}}^*, \to_{\mathcal{S}\cup Emb_{\mathcal{C}}}^+)$. Since $S \cup Emb_{\mathcal{C}}$ is terminating, Thm. 12 implies termination of $\mathcal{R} \cup Emb_{\mathcal{D}} \cup \mathcal{S} \cup Emb_{\mathcal{C}}$, i.e., simple termination of $\mathcal{R} \cup \mathcal{S}$. $\square$

The restriction to simple termination excludes many practically relevant TRSs. Thm. 13 illustrates that the size-change principle cannot compete with new techniques (e.g., *dependency pairs* [1] or the *monotonic semantic path ordering* [4]) where simplification orderings may be applied to non-simply terminating TRSs as well. However, these new techniques require methods to generate underlying base orderings. Hence, there is still an urgent need for powerful simplification orderings.

In the remainder of this section, we clarify the connection between size-change termination and classical simplification orderings and show that size-change termination and classical orderings do not subsume each other in general.

A major advantage of the size-change principle is that it can simulate the basic ingredients of RPOS, i.e., the concepts of *lexicographic* and of *multiset*-comparison. Thus, by the size-change principle w.r.t. a very simple reduction pair like the embedding ordering we obtain an automated method for termination analysis which avoids the search problems of RPOS and which can still capture the idea of comparing tuples of arguments lexicographically or as multisets. Thm. 14 shows that lexicographic orderings are simulated by the size-change principle.

Theorem 14 (Simulating Lexicographic Comparison) *Let $(\succsim, \succ)$ be a reduction pair and let π be a permutation of $1, \ldots, n$. We define an ordering $\succ_{lex}$ on n-tuples as $(s_1, \ldots, s_n) \succ_{lex} (t_1, \ldots, t_n)$ iff there is an $1 \leq i \leq n$ such that $s_{\pi(i)} \succ t_{\pi(i)}$ and $s_{\pi(j)} \succsim t_{\pi(j)}$ for all $j < i$. If $s_1^* \succ_{lex} t_1^*, \ldots, s_k^* \succ_{lex} t_k^*$ (where s_j^* and t_j^* denote n-tuples of terms), then the TRS $\{f(s_1^*) \to f(t_1^*), \ldots, f(s_k^*) \to f(t_k^*)\}$ is size-change terminating w.r.t. $(\succsim, \succ)$.*

Proof. All size-change graphs have edges $\pi(i)_f \xrightarrow{\succ} \pi(i)_f$ for some i and $\pi(j)_f \xrightarrow{\succsim} \pi(j)_f$ for all $j < i$. Concatenation of such graphs again yields a graph of this form and thus, all maximal multigraphs are also of this shape. Hence, they all contain an edge of the form $\pi(i)_f \xrightarrow{\succ} \pi(i)_f$ which proves size-change termination. $\square$

The construction in the proof is illustrated in Fig. 1. Here, the first size-change graph corresponds to a rule $f(s^*) \to f(t^*)$ where $s^* \succ_{lex} t^*$ holds and where the strict decrease is in the argument $\pi(i)$. The second graph has the strict decrease in argument $\pi(i')$. If $i \leq i'$, then their concatenation again results in a graph with strict decrease in argument $\pi(i)$.

Thus, size-change termination w.r.t. the same reduction pair $(\succsim, \succ)$ can simulate $\succ_{lex}$ for *any permutation* π used to compare the components of a tuple.

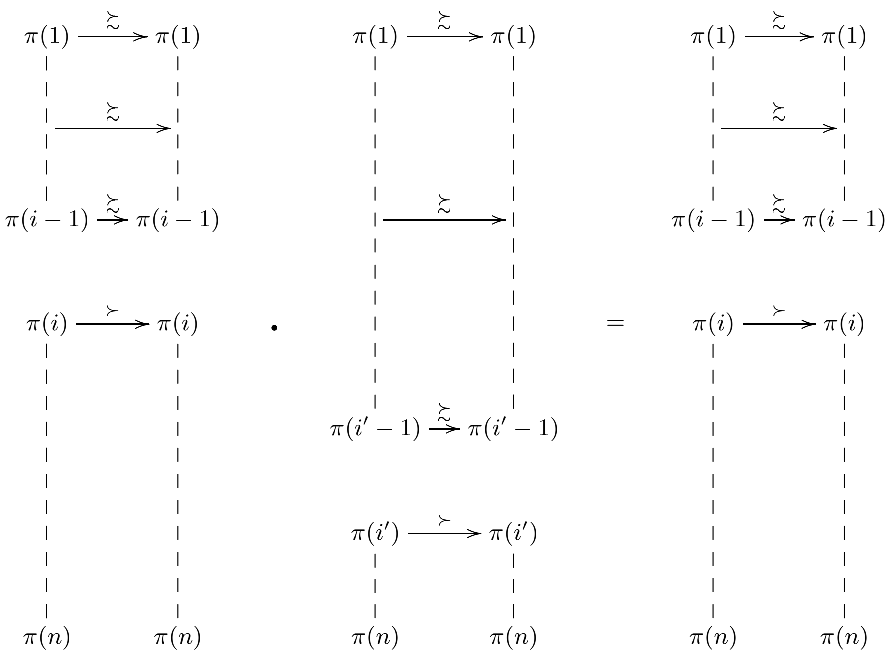


Fig. 1. Lexicographic Comparison with Size-Change Graphs

For example, regard the TRS $\{\mathsf{ack}(\mathsf{0},y) \to \mathsf{s}(y),\ \mathsf{ack}(\mathsf{s}(x),\mathsf{0}) \to \mathsf{ack}(x,\mathsf{s}(\mathsf{0})),\ \mathsf{ack}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{ack}(x,\mathsf{ack}(\mathsf{s}(x),y))\}$ computing the Ackermann function. The TRS is size-change terminating w.r.t. the embedding ordering on constructors, whereas with traditional term rewriting techniques, one would have to use the lexicographic path ordering. The next theorem shows that size-change termination can also simulate multiset comparison.

Theorem 15 (Simulating Multiset Comparison) *Let $(\succsim, \succ)$ be a reduction pair and let $\succ_{mul}$ compares tuples $(s_1, \dots, s_n)$ and $(t_1, \dots, t_n)$ by replacing at least one s_i by zero or more components t_j that are $\succ$-smaller than s_i. If $s_1^* \succ_{mul} t_1^*, \dots, s_k^* \succ_{mul} t_k^*$, then the TRS $\{f(s_1^*) \to f(t_1^*), \dots, f(s_k^*) \to f(t_k^*)\}$ is size-change terminating w.r.t. $(\succsim, \succ)$.*

Proof. In all size-change graphs, we can select a subset of edges with the following properties: (1) all input nodes have exactly one selected incoming edge, (2) for each output node, if one selects an outgoing edge labeled with "$\succsim$", then no other edge starting in this node may be selected, (3) at least one edge labeled with "$\succ$" is selected. It is easy to see that if one concatenates such size-change graphs G_1 and G_2 and selects those edges which result from the concatenation of two selected edges in G_1 and G_2, then the selected edges in the resulting multigraph also satisfy the conditions (1) – (3). Hence, the properties (1) – (3) also hold for the maximal multigraphs. Due to (3), there exists a selected edge $i_f \xrightarrow{\succ} j_f$ in each maximal multigraph. By (1), there is also a selected edge $k_f \to i_f$

reaching the input node marked with i_f. In the concatenation of the multigraph with itself, $k_f \to i_f \xrightarrow{\succsim} j_f$ would give rise to a (selected) edge $k_f \xrightarrow{\succsim} j_f$. Since maximal multigraphs are idempotent, the multigraph itself must already contain the (selected) edge $k_f \xrightarrow{\succsim} j_f$. Then (1) implies that $k_f = i_f$ and hence, we have a selected edge $k_f = i_f \to i_f$. Due to (2), this edge must be labeled with "$\succ$" and thus, size-change termination is proved. $\square$

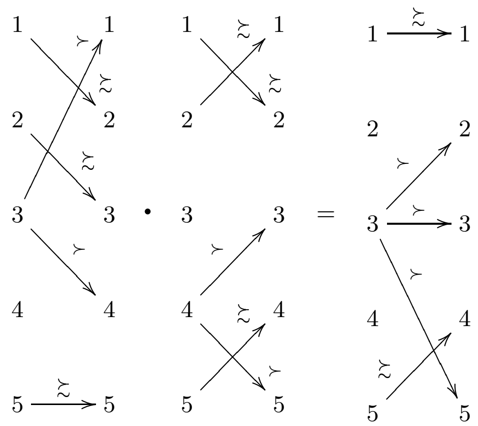


Fig. 2. Multiset Comparison with Size-Change Graphs

The construction in the proof is illustrated in Fig. 2, where we only depicted the *selected* edges of the graphs. Thus, every input node is reached by one unique edge (1) and every output node may have at most one outgoing "$\succsim$" edge (2). Moreover, there must be at least an "$\succ$" edge in the graph (3). The example in Fig. 2 demonstrates that the properties (1) – (3) are indeed preserved under concatenation of graphs.

For example, the TRS $\{\mathsf{plus}(\mathsf{0}, y) \to y,\ \mathsf{plus}(\mathsf{s}(x), y) \to \mathsf{s}(\mathsf{plus}(y, x))\}$ where plus permutes its arguments is size-change terminating w.r.t. the embedding ordering on constructors, whereas in existing rewriting approaches one would have to use the recursive (multiset) path ordering.

Since both lexicographic and multiset comparison are simulated by the size-change principle using the *same* reduction pair, one can also handle TRSs like Ex. 2 where traditional path orderings like RPOS (or KBO) fail. In the first rule $\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(x, \mathsf{s}(x))$ the arguments of f have to be compared lexicographically from left to right and in the second rule $\mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(y, x)$ they have to be compared as multisets. If one adds the rules for the Ackermann function then polynomial orderings fail as well, but size-change termination is proved as before.

However, compared to classical path orderings, the size-change principle also has several drawbacks. One problem is that it can only simulate lexicographic and multiset comparison for the arguments of the *root* symbol. Hence, if one adds a new function on top of all terms in the rules, this simulation is no longer possible.

For example, the TRS $\{\mathsf{f}(\mathsf{plus}(\mathsf{0}, y)) \to \mathsf{f}(y), \mathsf{f}(\mathsf{plus}(\mathsf{s}(x), y)) \to \mathsf{f}(\mathsf{s}(\mathsf{plus}(y, x)))\}$ is no longer size-change terminating w.r.t. the embedding ordering, whereas classical path orderings can apply lexicographic or multiset comparisons on all levels of the term. Thus, termination would still be easy to prove with RPO.

Perhaps the most serious drawback is that the size-change principle lacks concepts to compare defined function symbols syntactically. Consider a TRS with the rule $\mathsf{log}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{log}(\mathsf{s}(\mathsf{half}(x))))$ and rules for half such that $\mathsf{half}(x)$ computes $\lfloor \frac{x}{2} \rfloor$. If a function (like log) calls another defined function (like half) in the arguments of its recursive calls, one has to check whether the argument $\mathsf{half}(x)$ is smaller than the term $\mathsf{s}(x)$ in the corresponding left-hand side. The size-change principle on its own offers no possibility for that and its mechanizable versions (Thm. 9 and Thm. 12) fail since they only use an underlying ordering on constructor terms. In contrast, classical orderings like RPO can easily show termination automatically using a *precedence* $\mathsf{log} > \mathsf{s} > \mathsf{half}$ on function symbols.

Finally, the size-change principle has the disadvantage that it cannot *measure* terms by combining measures of subterms as in polynomial orderings or KBO.

Example 16 Term measures (or *weights*) are particularly useful if one parameter is increasing, but the decrease of another parameter is greater than this increase. So termination of $\{\mathsf{plus}(\mathsf{s}(\mathsf{s}(x)), y) \to \mathsf{s}(\mathsf{plus}(x, \mathsf{s}(y))),\ \mathsf{plus}(x, \mathsf{s}(\mathsf{s}(y))) \to \mathsf{s}(\mathsf{plus}(\mathsf{s}(x), y)), \mathsf{plus}(\mathsf{s}(\mathsf{0}), y) \to \mathsf{s}(y),\ \mathsf{plus}(\mathsf{0}, y) \to y\}$ is trivial to prove with polynomial orderings or KBO, but the TRS is not size-change terminating w.r.t. *any* reduction pair.

5 Comparison and Combination with Dependency Pairs

Now we compare the size-change principle with *dependency pairs*. In contrast to other recent techniques [4, 6], dependency pairs and size-change graphs are both built from recursive calls which suggests to combine these approaches to benefit from their respective advantages.

We briefly recapitulate the concepts of dependency pairs; see [1, 7, 8] for refinements and motivations. Let $\mathcal{F}^\sharp = \{f^\sharp \mid f \in \mathcal{D}\}$ be a set of *tuple symbols*, where $f^\sharp$ has the same arity as f and we often write F for $f^\sharp$, etc. If $t = g(t_1, \ldots, t_m)$ with $g \in \mathcal{D}$, we write $t^\sharp$ for $g^\sharp(t_1, \ldots, t_m)$. If $l \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol, then the rewrite rule $l^\sharp \to t^\sharp$ is called a *dependency pair* of $\mathcal{R}$. So the dependency pairs of the TRS from Ex. 2 are

$$\mathsf{F}(\mathsf{s}(x), y) \to \mathsf{F}(x, \mathsf{s}(x)) \quad (4) \qquad\qquad \mathsf{F}(x, \mathsf{s}(y)) \to \mathsf{F}(y, x) \quad (5)$$

We always assume that different occurrences of dependency pairs are variable disjoint. Then a TRS is (innermost) terminating iff there is no infinite (innermost) *chain* of dependency pairs. A sequence $s_1 \to t_1, s_2 \to t_2, \ldots$ of dependency pairs is a *chain* iff $t_i\sigma \to_{\mathcal{R}}^* s_{i+1}\sigma$ for all i and a suitable substitution σ. The sequence is an *innermost* chain iff $t_i\sigma \xrightarrow{\mathsf{i}}_{\mathcal{R}}^* s_{i+1}\sigma$ and all $s_i\sigma$ are in normal form.

To estimate which dependency pairs may occur consecutively in chains, one builds a so-called dependency graph. Let $\textsc{cap}(t)$ result from replacing all subterms of t with defined root symbol by different fresh variables and let $\text{REN}(t)$ result from replacing all occurrences of variables in t by different fresh variables. For instance, $\text{CAP}(\mathsf{F}(x, \mathsf{s}(x))) = \mathsf{F}(x, \mathsf{s}(x))$ and $\text{REN}(\mathsf{F}(x, \mathsf{s}(x))) = \mathsf{F}(x_1, \mathsf{s}(x_2))$. The (estimated) *dependency graph* is the directed graph whose nodes are the dependency pairs and there is an arc from $s \to t$ to $v \to w$ iff $\text{REN}(\text{CAP}(t))$ and v are unifiable. In the (estimated) *innermost dependency graph* there is only an arc from $s \to t$ to $v \to w$ iff $\text{CAP}(t)$ and v are unifiable. For the TRS of Ex. 2, the dependency graph and the innermost dependency graph are identical and each dependency pair is connected with itself and with the other pair.

A non-empty set $\mathcal{P}$ of dependency pairs is a *cycle* if for any pairs $s \to t$ and $v \to w$ in $\mathcal{P}$ there is a non-empty path from $s \to t$ to $v \to w$ which only traverses pairs from $\mathcal{P}$. In our example we have the cycles $\{(4)\}$, $\{(5)\}$, and $\{(4), (5)\}$. If a cycle only contains dependency pairs resulting from the rules $\mathcal{R}' \subseteq \mathcal{R}$ we speak of an *$\mathcal{R}'$-cycle* of the dependency graph of $\mathcal{R}$. Finally, for $f \in \mathcal{D}$ we define its *usable rules* $\mathcal{U}(f)$ as the smallest set containing all f-rules and all rules that are usable for function symbols occurring in right-hand sides of f-rules. In our example, the usable rules for f are (1) and (2). For $\mathcal{D}' \subseteq \mathcal{D}$ let $\mathcal{U}(\mathcal{D}') = \bigcup_{f \in \mathcal{D}'} \mathcal{U}(f)$.

Theorem 17 (Dependency Pair Approach [1]) *A TRS $\mathcal{R}$ is terminating iff for each cycle $\mathcal{P}$ in the dependency graph there is a monotonic reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ such that*

(a) $s \succsim t$ for all $s \to t \in \mathcal{P}$ and $s \succ t$ for at least one $s \to t \in \mathcal{P}$
(b) $l \succsim r$ for all $l \to r \in \mathcal{R}$.

$\mathcal{R}$ is innermost terminating if for each cycle $\mathcal{P}$ in the innermost dependency graph there is a monotonic reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ such that

(c) $s \succsim t$ for all $s \to t \in \mathcal{P}$ and $s \succ t$ for at least one $s \to t \in \mathcal{P}$
(d) $l \succsim r$ for all $l \to r \in \mathcal{U}(\mathcal{D}')$,
where $\mathcal{D}' = \{f \mid f \in \mathcal{D}$ occurs in t for some $s \to t \in \mathcal{P}\}$.

For the TRS in Ex. 2, in the cycle $\mathcal{P} = \{(4), (5)\}$ we have to find a reduction pair such that one dependency pair is weakly decreasing (w.r.t. $\succsim$) and the other is strictly decreasing (w.r.t. $\succ$). Since $\succ$ does not have to be monotonic, a key ingredient of the dependency pair approach is to use a standard simplification ordering in combination with an *argument filtering* which eliminates argument positions of function symbols. For example, we may eliminate the second argument position of F. In this way, F becomes unary and every term $\mathsf{F}(s, t)$ is replaced by $\mathsf{F}(s)$. Then the constraint $\mathsf{F}(\mathsf{s}(x)) \succ \mathsf{F}(x)$ resulting from the dependency pair (4) is easily satisfied but there is no reduction pair satisfying the constraint $\mathsf{F}(x) \succsim \mathsf{F}(y)$ from the second dependency pair (5). Indeed, there exists no argument filtering such that the constraints resulting from the dependency pair approach would be satisfied by a standard path ordering like RPOS or KBO. Moreover, if one adds the rules $\mathsf{f}(x, y) \to \mathsf{ack}(x, y)$, $\mathsf{ack}(\mathsf{s}(x), y) \to \mathsf{f}(x, x)$, and the rules for the Ackermann function ack, then the dependency pair constraints are not satisfied by any polynomial ordering either.

Thus, termination cannot be proved with dependency pairs in combination with classical orderings amenable to automation, whereas the proof is very easy with the size-change principle and a simple reduction pair like the embedding ordering on constructors. While the examples in [13] are easily handled by dependency pairs and RPOS, this shows that there exist TRSs where the size-change principle is preferable to dependency pairs and standard rewrite orderings.

In fact, size-change termination encompasses the concept of argument filtering for root symbols, since it concentrates on certain arguments of (root) function symbols while ignoring others. This is an advantage compared to dependency pairs where finding the argument filtering is a major search problem. Moreover, the size-change principle examines sequences of function calls in a more sophisticated way. Depending on the different "paths" from one function call to another, it can choose different arguments to be (strictly) decreasing. In contrast, in the dependency pair approach such choices remain fixed for the whole cycle.

On the other hand, in addition to the drawbacks mentioned in Sect. 4, a disadvantage of the size-change principle is that it is not modular, i.e., one has to use the same reduction pair for the whole termination proof whereas the dependency pair approach permits different orderings for different cycles. The size-change principle also does not analyze arguments of terms to check whether two function calls can follow each other, whereas in dependency graphs, this is approximated using the functions CAP and REN. Again, the most severe drawback is that the size-change principle offers no technique to compare terms with defined symbols, whereas dependency pairs use inequalities of the form $l \succsim r$ for this purpose. For that reason, only very restricted reduction pairs may be used for the size-change principle in Thm. 9 and 12, whereas one may use arbitrary monotonic reduction pairs for the dependency pair approach. In fact, dependency pairs are a *complete* technique which can prove termination of every TRS, whereas this is not at all true for the size-change principle (see e.g., Ex. 16).

In the remainder, we introduce a new technique to combine dependency pairs and size-change termination. A straightforward approach would be to use dependency pairs as a preprocessing step and size-change termination as the "base ordering" when trying to satisfy the constraints resulting from the dependency pair approach. However, this would be very weak due to the restrictions on the reduction pairs in Thm. 9 and Thm. 12.

Instead, we incorporate the size-change principle into the dependency pair approach and use it when generating the constraints. The resulting technique is stronger than both previous approaches: If (innermost) termination can be proved by the size-change principle or by dependency pairs using certain reduction pairs, then it can also be proved with our new technique using the *same* reduction pairs. On the other hand, there are many examples which cannot be proved by the size-change principle and where dependency pairs would require complicated reduction pairs (that can hardly be generated automatically), whereas with our combined technique the (automatic) proof works with very simple reduction pairs, cf. the appendix.

Obviously, size-change graphs and dependency pairs have a close correspondence, since they both represent a call of a defined symbol g in the right-hand side of a rewrite rule $f(s_1,\ldots,s_n) \to \ldots g(t_1,\ldots,t_m)\ldots$ Since we only need to concatenate size-change graphs which correspond to cycles in the (innermost) dependency graph, we now label size-change graphs by the corresponding dependency pair and multigraphs are labeled by the corresponding sequence of dependency pairs. Then two size-change graphs or multigraphs labeled with $(\ldots,D)$ and $(D',\ldots)$ may only be concatenated if there is an arc from D to D' in the (innermost)[1] dependency graph. Another problem is that in size-change graphs one only has output nodes $1_f,\ldots,n_f$ and input nodes $1_g,\ldots,m_g$ to compare the *arguments* of f and g. Therefore, the size-change principle cannot deal with TRSs like Ex. 16 where one has to regard the *whole* term in order to show termination. For that reason we add another output node ϵ_f and input node ϵ_g which correspond to the whole terms (or more precisely, to the terms $F(s_1,\ldots,s_n)$ and $G(t_1,\ldots,t_m)$ of the corresponding dependency pair).

Definition 18 (Extended Size-Change Graphs) *Let $(\succsim,\succ)$ be a reduction pair on $\mathcal{T}(\mathcal{F}\cup\mathcal{F}^\sharp,\mathcal{V})$. For every rule $f(s_1,\ldots,s_n) \to r$ of a TRS $\mathcal{R}$ and every subterm $g(t_1,\ldots,t_m)$ of r with $g \in \mathcal{D}$, the* extended size-change graph *has $n+1$ output nodes i_f and $m+1$ input nodes j_g where $i \in \{\epsilon,1,\ldots,n\}$, $j \in \{\epsilon,1,\ldots,m\}$. Let $s = F(s_1,\ldots,s_n)$ and $t = G(t_1,\ldots,t_m)$. Then there is an edge $i_f \xrightarrow{\succ} j_g$ iff $s|_i \succ t|_j$ and otherwise, there is an edge $i_f \xrightarrow{\succsim} j_g$ iff $s|_i \succsim t|_j$. Moreover, every extended size-change graph is labeled by a one-element sequence $(F(s_1,\ldots,s_n) \to G(t_1,\ldots,t_m))$.*

Concatenation of extended size-change graphs to extended multigraphs works as in Def. 3. However, if G is a multigraph labeled with $(D_1,\ldots,D_n)$ and H is labeled with $(D'_1,\ldots,D'_m)$, then they can only be concatenated if there is an arc from D_n to D'_1 in the (innermost) dependency graph. The concatenation $G \bullet H$ is labeled with $(D_1,\ldots,D_n,D'_1,\ldots,D'_m)$.

As an example, reconsider the TRS for the Ackermann function. The rule $\mathsf{ack}(\mathsf{s}(x),\mathsf{0}) \to \mathsf{ack}(x,\mathsf{s}(\mathsf{0}))$ gives rise to the following extended size-change graph if we use the embedding ordering on constructors.

$$\begin{tikzcd}
\epsilon_{\mathsf{ack}} \arrow[dr, "\succ"] & \epsilon_{\mathsf{ack}} \\
1_{\mathsf{ack}} \arrow[r, "\succ"'] & 1_{\mathsf{ack}} \\
2_{\mathsf{ack}} & 2_{\mathsf{ack}}
\end{tikzcd}$$

This graph is labeled with the singleton sequence consisting of the dependency pair $\mathsf{ACK}(\mathsf{s}(x),\mathsf{0}) \to \mathsf{ACK}(x,\mathsf{s}(\mathsf{0}))$. Thus, it cannot be concatenated with itself,

[1] Whether one regards the dependency graph or the innermost dependency graph depends on whether one wants to prove termination or innermost termination.

since there is no arc from this dependency pair to itself in the (innermost) dependency graph.

In the remainder, when we speak of size-change graphs or multigraphs, we always mean *extended* graphs. Obviously, there may exist infinitely many multigraphs due to the labeling with a sequence of dependency pairs. However, two multigraphs with labelings $(D_1, \ldots, D_n)$ and $(D'_1, \ldots, D'_m)$ are identified if their nodes and edges are identical and if $D_1 = D'_1$, $D_n = D'_m$, and $\{D_1, \ldots, D_n\} = \{D'_1, \ldots, D'_m\}$. Thus, for the labeling only the set of dependency pairs and the first and last dependency pair of the sequences is important. Then, there are again only finitely many different multigraphs.

To combine dependency pairs and the size-change principle now we only regard multigraphs labeled with a cycle $\mathcal{P}$ of the (innermost) dependency graph (i.e., they are labeled with $(D_1, \ldots, D_n)$ such that $\mathcal{P} = \{D_1, \ldots, D_n\}$). Moreover, one may use different reduction pairs for the multigraphs resulting from different cycles. To benefit from the advantages of the size-change principle (i.e., combining lexicographic and multiset comparison and using different argument filterings and strict inequalities within one cycle), we do not build inequalities but size-change graphs out of the dependency pairs.

The following theorem combines dependency pairs and the size-change principle for full termination (Thm. 12). In contrast to Thm. 12 we now allow arbitrary reduction pairs. However, to handle defined symbols properly, one then has to require that all rules are weakly decreasing (like in the dependency pair approach). Alternatively, as in Thm. 12 one may also use reduction pairs $(\to^*_{\mathcal{S}}, \to^+_{\mathcal{S}})$ for a terminating non-duplicating TRS $\mathcal{S}$ over the constructors of $\mathcal{R}$ without requiring that $\mathcal{R}$'s rules are weakly decreasing. For example, in this way one can prove termination of the Ackermann TRS with the embedding ordering (i.e., $\mathcal{S} = Emb_{\mathcal{C}}$). However, in order to use $(\to^*_{\mathcal{S}}, \to^+_{\mathcal{S}})$ for some cycles and other reduction pairs $(\succsim, \succ)$ for other cycles, one has to prove termination of $\mathcal{R} \cup \mathcal{S}$ instead of just $\mathcal{R}$.

Example 19 To illustrate this, let $\mathcal{R} = \{\mathsf{g}(\mathsf{f}(\mathsf{a})) \to \mathsf{g}(\mathsf{f}(\mathsf{b})), \mathsf{f}(\mathsf{b}) \to \mathsf{f}(\mathsf{a})\}$ and $\mathcal{S} = \{\mathsf{a} \to \mathsf{b}\}$. The only cycle of $\mathcal{R}$'s dependency graph is $\{\mathsf{G}(\mathsf{f}(\mathsf{a})) \to \mathsf{G}(\mathsf{f}(\mathsf{b}))\}$ and for this cycle, size-change termination can be shown using $(\to^*_{\mathcal{S}}, \to^+_{\mathcal{S}})$. Thus, if one only regards $\mathcal{R}$ instead of $\mathcal{R} \cup \mathcal{S}$, one could falsely "prove" termination of $\mathcal{R}$. Instead, $\{\mathsf{F}(\mathsf{b}) \to \mathsf{F}(\mathsf{a})\}$ must also be regarded, since it is an $\mathcal{R}$-cycle of the dependency graph of $\mathcal{R} \cup \mathcal{S}$ (because in $\mathcal{R} \cup \mathcal{S}$, a is a defined symbol). Moreover, for reduction pairs $(\succsim, \succ) \neq (\to^*_{\mathcal{S}}, \to^+_{\mathcal{S}})$, one has to demand $l \succsim r$ not only for the rules $l \to r$ of $\mathcal{R}$, but for those of $\mathcal{S}$ as well. Otherwise, the constraints for the cycle $\{\mathsf{F}(\mathsf{b}) \to \mathsf{F}(\mathsf{a})\}$ would falsely be satisfiable.

By Thm. 20, the resulting termination criterion is sound, complete, and more powerful than the size-change principle or dependency pairs on their own.

Theorem 20 (Termination Proofs) *Let $\mathcal{R}$ be a TRS over $\mathcal{F}$ with constructors $\mathcal{C}$ and let $\mathcal{S}$ be a terminating non-duplicating TRS over $\mathcal{C}$. $\mathcal{R}$ (and even $\mathcal{R} \cup \mathcal{S}$) is terminating iff for each $\mathcal{R}$-cycle $\mathcal{P}$ in the dependency graph of $\mathcal{R} \cup \mathcal{S}$ there is a monotonic reduction pair $(\succsim, \succ)$ on $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ such that*

(a) all maximal multigraphs w.r.t. $(\succsim, \succ)$ labeled with $\mathcal{P}$ contain an edge $i \xrightarrow{\succ} i$
(b) $\succsim = \to_{\mathcal{S}}^*$ and $\succ = \to_{\mathcal{S}}^+$ or $l \succsim r$ for all $l \to r \in \mathcal{R} \cup \mathcal{S}$

If $\mathcal{R}$ is size-change terminating w.r.t. $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$ as in Thm. 12 or if a reduction pair satisfies Conditions (a) and (b) of Thm. 17 for termination with dependency pairs, then this reduction pair also satisfies the conditions of this criterion.

Proof. The above criterion can simulate size-change termination (Thm. 12): If every maximal multigraph contains an edge $i \xrightarrow{\succ} i$ then this also holds for those maximal multigraphs that are labeled with $\mathcal{P}$. It can also simulate dependency pairs by choosing $\mathcal{S} = \varnothing$: Condition (a) in Thm. 17 implies that every multigraph labeled with $\mathcal{P}$ must contain the edge $\epsilon \xrightarrow{\succ} \epsilon$. Since the dependency pair approach is *complete* for termination (even with estimated or no dependency graphs), this also proves the "only if" direction.

For the "if" direction, suppose that $\mathcal{R} \cup \mathcal{S}$ is not terminating. Since $\mathcal{S}$ terminates, by Lemma 11 and the soundness of dependency pairs, there is an infinite chain $s_1 \to t_1, s_2 \to t_2, \ldots$ of $\mathcal{R}$-dependency pairs such that $t_i\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}\sigma$ for all i and a substitution σ, and $s_1 = s^\sharp$ for a minimal non-terminating term s w.r.t. $\mathcal{R} \cup \mathcal{S}$. Moreover, there is an $\mathcal{R}$-cycle $\mathcal{P}$ consisting of those dependency pairs which occur infinitely often in this chain. Let $i_1 < i_2 < \ldots$ such that $\{s_{i_j} \to t_{i_j}, \ldots, s_{i_{j+1}-1} \to t_{i_{j+1}-1}\} = \mathcal{P}$ for all j, i.e., we partition the sequence into parts where all dependency pairs of $\mathcal{P}$ occur. For all j, let G_j be the multigraph resulting from the concatenation of the size-change graphs corresponding to $s_{i_j} \to t_{i_j}, \ldots, s_{i_{j+1}-1} \to t_{i_{j+1}-1}$. Note that all G_j are labeled with $\mathcal{P}$.

Due to (a), every multigraph H resulting from concatenation of size-change graphs contains an edge of the form $i \xrightarrow{\succ} i$, provided that $H = H \bullet H$ and that H is labeled with $\mathcal{P}$. Hence, every idempotent multigraph $H = H \bullet H$ resulting from concatenating graphs from $G_1, G_2, \ldots$ also contains an edge $i \xrightarrow{\succ} i$. The reason is that since all G_j are labeled with $\mathcal{P}$, then H is also labeled with $\mathcal{P}$.

From this, Lemma 7 implies that there is an infinite path with infinitely many "$\succ$"-edges in the infinite graph resulting from $G_1, G_2, \ldots$ by identifying the input nodes of G_j with the output nodes of G_{j+1}. Hence, there is also such an infinite path in the infinite graph resulting from the size-change graphs corresponding to $s_1 \to t_1, s_2 \to t_2, \ldots$ Without loss of generality, we assume that the infinite path already starts in the size-change graph corresponding to $s_1 \to t_1$. For every i, let a_i be the output node in the size-change graph of $s_i \to t_i$ which is on this path. For infinitely many i we have $s_i|_{a_i}\sigma \succ t_i|_{a_{i+1}}\sigma$ and otherwise, we have $s_i|_{a_i}\sigma \succsim t_i|_{a_{i+1}}\sigma$, since $\succsim$ and $\succ$ are closed under substitutions.

If the reduction pair $(\succsim, \succ)$ is $(\to_{\mathcal{S}}^*, \to_{\mathcal{S}}^+)$, then we obtain a contradiction to the minimality of s similar as in the proof of Thm. 12. Otherwise, $t_i|_{a_{i+1}}\sigma \succsim s_{i+1}|_{a_{i+1}}\sigma$ due to (b) since $t_i|_{a_{i+1}}\sigma \to_{\mathcal{R}\cup\mathcal{S}}^* s_{i+1}|_{a_{i+1}}\sigma$. Hence, we have an infinite decreasing sequence w.r.t. $\succ$ which contradicts its well-foundedness. $\square$

In the corresponding approach for innermost termination, we integrate the technique of Thm. 9 with dependency pairs. (Integrating a variant of Thm. 12

for innermost termination would have the disadvantage that one would prove innermost termination of $\mathcal{R} \cup \mathcal{S}$ which does not imply innermost termination of $\mathcal{R}$.) In the dependency pair approach for innermost termination, only the *usable* rules for defined symbols in right-hand sides t of dependency pairs $s \to t$ have to be weakly decreasing. Here, one can benefit from the size-change principle, which restricts the comparison of terms to certain arguments. Function symbols of t which do not occur in the arguments being compared do not have to be regarded as being "usable". More precisely, if one uses the extension of a reduction pair which only compares terms with defined symbols from a subset $\mathcal{D}' \subseteq \mathcal{D}$, then one only has to require weak decreasingness of $\mathcal{U}(\mathcal{D}')$. Thus, here the size-change principle has the important advantage that one can reduce the set of usable rules.

For example, the Ackermann TRS has the rule $\mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{ack}(\mathsf{s}(x), y))$ and therefore, we obtain the dependency pair $\mathsf{ACK}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ACK}(x, \mathsf{ack}(\mathsf{s}(x), y))$. Since ack occurs in the right-hand side of this dependency pair, in the dependency pair approach we would have to require $l \succsim r$ for all ack-rules since they would be regarded as being usable. For this reason, we would need a lexicographic comparison. However, in our new technique, the ACK-dependency pairs are transformed into size-change graphs and size-change termination can easily be shown using the embedding ordering on constructor terms (i.e., $\mathcal{D}' = \varnothing$). In other words, the second argument of $\mathsf{ACK}(x, \mathsf{ack}(\mathsf{s}(x), y))$ is never regarded in this comparison and therefore, the ack-rules are no longer usable. So instead of LPO we only need the embedding ordering to satisfy the resulting constraints. Hence, in the combined technique one can often use much simpler reduction pairs than the reduction pairs needed with dependency pairs.

Here it is important that extensions are non-monotonic. Consider the TRS of Ex. 19 and a reduction pair on constructor terms (i.e., $\mathcal{D}' = \varnothing$) where a is greater than b. Hence, we do not have to regard any usable rules. In the extension $(\succsim, \succ)$ of this reduction pair we have $\mathsf{f}(\mathsf{a}) \not\succ \mathsf{f}(\mathsf{b})$. Thus, the dependency pair $\mathsf{G}(\mathsf{f}(\mathsf{a})) \to \mathsf{G}(\mathsf{f}(\mathsf{b}))$ is not decreasing, i.e., innermost termination is not proved. But if the extension were monotonic, we would falsely prove innermost termination of $\mathcal{R}$.

Theorem 21 (Innermost Termination Proofs) *A TRS $\mathcal{R}$ is innermost terminating if for each cycle $\mathcal{P}$ in the innermost dependency graph there is a reduction pair on $\mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$* [2] *for some $\mathcal{D}' \subseteq \mathcal{D}$ which is monotonic if $\mathcal{D}' \neq \varnothing$, such that for its extension $(\succsim, \succ)$ to $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V})$ we have*

(a) all maximal multigraphs w.r.t. $(\succsim, \succ)$ labeled with $\mathcal{P}$ contain an edge $i \xrightarrow{\succ} i$
(b) $l \succsim r$ for all $l \to r \in \mathcal{U}(\mathcal{D}')$

[2] Instead of a reduction pair on $\mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$ one can also use a reduction pair $(\succsim', \succ')$ with $\succsim', \succ' \subseteq \mathcal{T}(\mathcal{F} \cup \mathcal{F}^\sharp, \mathcal{V}) \times \mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$. Here, $\succsim'$ and $\succ'$ must be closed under substitutions with terms from $\mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$ and the reduction pair is considered to be monotonic if $\succsim'$ is closed under $\mathcal{T}(\mathcal{C} \cup \mathcal{D}' \cup \mathcal{F}^\sharp, \mathcal{V})$-contexts. The advantage of this modification is that one can deal with defined symbols on left-hand sides of dependency pairs without including them in $\mathcal{D}'$.

If $\mathcal{R}$ is size-change terminating w.r.t. a reduction pair as in Thm. 9 or if a reduction pair satisfies Conditions (c) and (d) of Thm. 17 for innermost termination with dependency pairs, then it also satisfies the conditions of this criterion.

Proof. Thm. 21 can simulate the size-change principle: As in Thm. 20, size-change termination implies (a). Moreover, if $(\succsim, \succ)$ is the extension of a reduction pair on $\mathcal{T}(\mathcal{C}, \mathcal{V})$ as in Thm. 9, then $\mathcal{D}' = \varnothing$ and thus, (b) is also satisfied.

The simulation of dependency pairs and the soundness of the above criterion are shown as for Thm. 20. If $\mathcal{R}$ is not innermost terminating, then there is an infinite innermost chain $s_1 \to t_1, s_2 \to t_2, \ldots$ with $t_i\sigma \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}} s_{i+1}\sigma$ and all $s_i\sigma$ are normal forms. As in Thm. 20's proof, this implies that in the infinite graph resulting from the corresponding size-change graphs there is an infinite path with infinitely many "$\succ$" labels. For every i, let a_i be the output node in the size-change graph corresponding to $s_i \to t_i$ which is on this infinite path. To conclude $t_i|_{a_{i+1}}\sigma \succsim s_{i+1}|_{a_{i+1}}\sigma$, note that $s_i|_{a_i} \succsim t_i|_{a_{i+1}}$ or $s_i|_{a_i} \succ t_i|_{a_{i+1}}$. According to the definition of extending reduction pairs, all subterms of $t_i|_{a_{i+1}}$ with root from $\mathcal{D} \setminus \mathcal{D}'$ also occur in $s_i|_{a_i}$. Hence, when instantiated by σ they are in normal form. Therefore, the only rules applicable to $t_i|_{a_{i+1}}\sigma$ are from $\mathcal{U}(\mathcal{D}')$. Moreover, above the redexes of $t_i|_{a_{i+1}}\sigma$ there are no symbols from $\mathcal{D} \setminus \mathcal{D}'$, since otherwise these redexes would also occur in the normal form $s_i|_{a_i}\sigma$. Now (b) ensures $t_i|_{a_{i+1}}\sigma \succsim s_{i+1}|_{a_{i+1}}\sigma$. The remainder is as in Thm. 20's proof. □

The combined technique can handle TRSs where both original techniques fail, since some rules require a lexicographic or multiset comparison and others require polynomial orderings. In the combined technique, a lexicographic or multiset comparison is implicit since the size-change principle is incorporated. Thus, the resulting constraints are often satisfied by simple polynomial orderings. For example, we unite the plus-TRS (Ex. 16) with the TRS for Ackermann's function, where $\mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{ack}(\mathsf{s}(x), y))$ is replaced by $\mathsf{ack}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{ack}(x, \mathsf{plus}(y, \mathsf{ack}(\mathsf{s}(x), y)))$. In the original dependency pair approach, both the ack- and plus-rules are usable for the corresponding dependency pair and thus, no standard ordering amenable to automation fulfills the resulting constraints. But in the combined technique, there are no usable rules and hence, the innermost termination proof works with the simple polynomial ordering on constructors and tuple symbols where $\mathsf{s}(x)$ is mapped to $x + 1$ and $\mathsf{PLUS}(x, y)$ is mapped to $x + y$. In practice, there are many TRSs where the combined technique simplifies the termination proof significantly (e.g., TRSs for arithmetic operations, for sorting algorithms, for term manipulations in λ-calculus, etc., cf. the appendix).

Example 22 To demonstrate the power of combining dependency pairs and the size-change principle, we consider the following TRS for sorting lists taken from [2, Ex. 3.10].

$$\mathsf{eq}(\mathsf{0}, \mathsf{0}) \to \mathsf{true} \tag{6}$$
$$\mathsf{eq}(\mathsf{0}, \mathsf{s}(x)) \to \mathsf{false} \tag{7}$$

$$\mathsf{eq}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{false} \tag{8}$$
$$\mathsf{eq}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{eq}(x, y) \tag{9}$$
$$\mathsf{le}(\mathsf{0}, y) \to \mathsf{true} \tag{10}$$
$$\mathsf{le}(\mathsf{s}(x), \mathsf{0} \to \mathsf{false} \tag{11}$$
$$\mathsf{le}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{le}(x, y) \tag{12}$$
$$\mathsf{app}(\mathsf{nil}, y) \to y \tag{13}$$
$$\mathsf{app}(\mathsf{add}(n, x), y) \to \mathsf{add}(n, \mathsf{app}(x, y)) \tag{14}$$
$$\mathsf{min}(\mathsf{add}(n, \mathsf{nil})) \to n \tag{15}$$
$$\mathsf{min}(\mathsf{add}(n, \mathsf{add}(m, x))) \to \mathsf{if}_{\mathsf{min}}(\mathsf{le}(n, m), \mathsf{add}(n, \mathsf{add}(m, x))) \tag{16}$$
$$\mathsf{if}_{\mathsf{min}}(\mathsf{true}, \mathsf{add}(n, \mathsf{add}(m, x))) \to \mathsf{min}(\mathsf{add}(n, x)) \tag{17}$$
$$\mathsf{if}_{\mathsf{min}}(\mathsf{false}, \mathsf{add}(n, \mathsf{add}(m, x))) \to \mathsf{min}(\mathsf{add}(m, x)) \tag{18}$$
$$\mathsf{rm}(n, \mathsf{nil}) \to \mathsf{nil} \tag{19}$$
$$\mathsf{rm}(n, \mathsf{add}(m, x)) \to \mathsf{if}_{\mathsf{rm}}(\mathsf{eq}(n, m), n, \mathsf{add}(m, x)) \tag{20}$$
$$\mathsf{if}_{\mathsf{rm}}(\mathsf{true}, n, \mathsf{add}(m, x)) \to \mathsf{rm}(n, x) \tag{21}$$
$$\mathsf{if}_{\mathsf{rm}}(\mathsf{false}, n, \mathsf{add}(m, x)) \to \mathsf{add}(m, \mathsf{rm}(n, x)) \tag{22}$$
$$\mathsf{minsort}(\mathsf{nil}, \mathsf{nil}) \to \mathsf{nil} \tag{23}$$
$$\mathsf{minsort}(\mathsf{add}(n, x), y) \to \mathsf{if}_{\mathsf{minsort}}(\mathsf{eq}(n, \mathsf{min}(\mathsf{add}(n, x))), \mathsf{add}(n, x), y) \tag{24}$$
$$\mathsf{if}_{\mathsf{minsort}}(\mathsf{true}, \mathsf{add}(n, x), y) \to \mathsf{add}(n, \mathsf{minsort}(\mathsf{app}(\mathsf{rm}(n, x), y), \mathsf{nil})) \tag{25}$$
$$\mathsf{if}_{\mathsf{minsort}}(\mathsf{false}, \mathsf{add}(n, x), y) \to \mathsf{minsort}(x, \mathsf{add}(n, y)) \tag{26}$$

To increase efficiency when using the dependency pair approach, instead of searching for different reduction pairs for every cycle, one often tries to use the same reduction pair $(\succsim, \succ)$ for all cycles in a strongly connected component (SCC) of the (estimated) dependency graph. Thus, instead of Constraint (a) in Thm. 17 we require the following constraints for all SCCs $\mathcal{P}$:

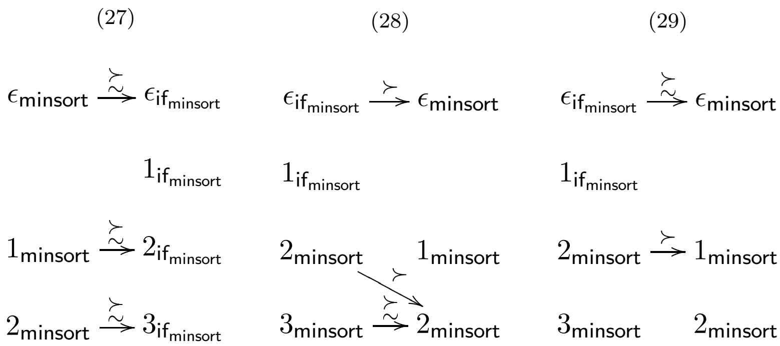
(a)$_1$ $s \succsim t$ for all $s \to t \in \mathcal{P}$
(a)$_2$ $s \succ t$ for at least one $s \to t \in \mathcal{P}'$ for each cycle $\mathcal{P}' \subseteq \mathcal{P}$

The most interesting part is to show the termination of $\mathsf{minsort}$ and $\mathsf{if}_{\mathsf{minsort}}$. The corresponding SCC consists of the following three dependency pairs.

$$\mathsf{MINSORT}(\mathsf{add}(n, x), y) \to \mathsf{IF}_{\mathsf{minsort}}(\mathsf{eq}(n, \mathsf{min}(\mathsf{add}(n, x))), \mathsf{add}(n, x), y) \tag{27}$$
$$\mathsf{IF}_{\mathsf{minsort}}(\mathsf{true}, \mathsf{add}(n, x), y) \to \mathsf{MINSORT}(\mathsf{app}(\mathsf{rm}(n, x), y), \mathsf{nil}) \tag{28}$$
$$\mathsf{IF}_{\mathsf{minsort}}(\mathsf{false}, \mathsf{add}(n, x), y) \to \mathsf{MINSORT}(x, \mathsf{add}(n, y)) \tag{29}$$

In order to prove the absence of infinite chains built from (27), (28), (29), one can show that in each cycle either the sum of both list sizes is reduced or the sum remains equal and the first list is shortened. The list sizes can be expressed in simple linear polynomials, but the lexicographic combination cannot be expressed with simple polynomials. Therefore in [2], polynomials of degree 2 have been used to simulate the lexicographic comparison.

In contrast to this, with the combined approach of Thm. 20 we are not forced to use complex polynomials, even when regarding SCCs instead of cycles. The reason is that the lexicographic combination can be simulated in the size-change graphs. We map $\mathsf{add}(n, x)$ to $n + x + 1$, $\mathsf{0}$, true, false, nil, eq, le are mapped to 0, $\mathsf{rm}(x, y)$ and $\mathsf{if}_{\mathsf{rm}}(b, x, y)$ are mapped to y, $\mathsf{min}(x)$, $\mathsf{if}_{\mathsf{min}}(b, x)$ are mapped to x, and $\mathsf{app}(x, y)$, $\mathsf{minsort}(x, y)$, $\mathsf{MINSORT}(x, y)$, $\mathsf{if}_{\mathsf{minsort}}(b, x, y)$, $\mathsf{IF}_{\mathsf{minsort}}(b, x, y)$ are mapped to $x + y$. Then all constraints from the rules can be oriented (i.e., $l \succsim r$ for all rules $l \to r$) and we obtain the following three size-change graphs:[3]



It is now easy to see that all maximal multigraphs either contain a $\succ$-edge between their ϵ-nodes or there is a $\succsim$-edge between the ϵ-nodes, and a $\succ$-edge between the but-last argument nodes.

The same advantage can be seen in Ex. 3.13 of [2] which computes a reachability predicate in directed graphs. Again, quadratic polynomials were used in [2] to integrate lexicographic effects into a polynomial ordering. And similarly, with the approach combining dependency pairs and the size-change principle, we can show the termination using simple linear polynomials.

In [1, 7], several refinements to manipulate dependency pairs by narrowing, rewriting, and instantiation were proposed. These refinements directly carry over to our combined technique. To summarize, the combination of dependency pairs and the size-change principle has two main advantages: First, one can now prove (innermost) termination of TRSs automatically where up to now an automated proof was impossible. Second, for many TRSs where up to now the termination proof required complicated reduction pairs involving a large search space, one can now use much simpler orderings which increases efficiency.

6 Conclusion

In this paper, we extended the size-change principle to prove (innermost) termination of arbitrary TRSs. Then we compared this principle with classical sim-

[3] To improve readability we did not depict all edges.

plification orderings from term rewriting: It is also restricted to proving simple termination, it incorporates lexicographic and multiset comparison for root symbols (although not below the root), but it cannot handle defined symbols or term measures and weights. Nevertheless, there are even examples where the size-change principle is advantageous to dependency pairs, since it can simulate argument filtering for root symbols and it can investigate how the size of arguments changes in sequences of subsequent function calls. On the other hand, the size-change principle is not modular and it lacks a concept like the dependency graph to analyze which function calls can follow each other. For that reason, we developed a new approach which combines the size-change principle with dependency pairs. This combined approach is more powerful than both previous techniques and it has the advantage that it often succeeds with much simpler argument filterings and base orderings than the dependency pair approach. We have implemented both the original dependency pair approach and the combined approach in the system AProVE and found that this combination often increases efficiency dramatically. With this combination and with an underlying reduction pair based on the lexicographic path ordering, 103 of the 110 examples in the collection of [2] could be proved innermost terminating fully automatically. Most of these proofs took less than a second and the longest proof took about 10 seconds. The remaining 7 examples in [2] only fail because of the underlying reduction pair (e.g., one would need polynomial orderings or KBO). More details on these experiments can be found in the appendix.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. T. Arts and J. Giesl. A collection of examples for termination of term rewriting using dependency pairs. Technical Report AIB-2001-09, RWTH Aachen, Germany, 2001. Available from `http://aib.informatik.rwth-aachen.de`.
3. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
4. C. Borralleras, M. Ferreira, and A. Rubio. Complete monotonic semantic path orderings. In *Proc. 17th CADE*, LNAI 1831, pages 346–364, 2000.
5. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.
6. O. Fissore, I. Gnaedig, and H. Kirchner. Induction for termination with local strategies. In *Proc. 4th International Workshop on Strategies in Automated Deduction*, ENTCS 58, 2001.
7. J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Applicable Algebra in Engineering, Communication and Computing*, 12(1,2):39–72, 2001.
8. J. Giesl, T. Arts, and E. Ohlebusch. Modular termination proofs for rewriting using dependency pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.
9. S. Kamin and J. J. Lévy. Two generalizations of the recursive path ordering. Unpublished Manuscript, University of Illinois, IL, USA, 1980.
10. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon, 1970.
11. K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In *Proc. 1st PPDP*, LNCS 1702, pages 48–62, 1999.

12. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.
13. C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Proc. POPL '01*, pages 81–92, 2001.
14. Y. Toyama. Counterexamples to the termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.

A Implementation and Experiments

We have developed a system AProVE (<u>A</u>utomated <u>Pro</u>gram <u>V</u>erification <u>E</u>nvironment) for mechanized verification of functional programs and term rewrite systems. To perform automated termination or innermost termination proofs of TRSs, the system currently offers the techniques of *recursive path orderings* (possibly with status) and *dependency pairs* (including recent refinements such as *narrowing*, *rewriting*, and *instantiation* of dependency pairs [7]). The tool is written in Java and termination proofs can be performed via a graphical user interface.

To evaluate the results developed in the present paper, we extended the system by an implementation of the size-change principle and by an automatic technique to prove innermost termination based on Thm. 21, i.e., on our combination of dependency pairs with the size-change principle. The combined technique was tested against the original dependency pair technique using the large collection of examples in [2].

We first present our algorithm to verify innermost termination of a TRS $\mathcal{R}$ with defined symbols $\mathcal{D}$ and give a detailed explanation afterwards:

1. Compute the (estimated) innermost dependency graph of $\mathcal{R}$.
2. For each strongly connected component $\mathcal{P}$ in the graph:
   2.1. Let $\mathcal{C}_\mathcal{P}$ be the set of the constructors occurring in $\mathcal{P}$,
   let $\mathcal{D}_\mathcal{P}$ be a subset of the defined symbols
   occurring in right-hand sides in $\mathcal{P}$,
   let π be an argument filtering over the signature $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}$.
   If all such $\mathcal{D}_\mathcal{P}$ and all argument filterings π on $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}$
   have already been examined without success,
   then abort with "`No Success`".
   2.2. Let $s \succ t$ iff $t \in \mathcal{T}(\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}, \mathcal{V})$ and $\pi(s) \rightarrow^+_{Emb_\mathcal{F}} \pi(t)$.
   Let $s \succsim t$ iff $s \succ t$ or $s = t$.
   2.3. Try to show that all maximal multigraphs w.r.t. $(\succsim, \succ)$
   labeled with $\mathcal{P}$ contain an edge $i \xrightarrow{\succ} i$.
   2.4. If Step 2.3 fails, then go to Step 2.1 and
   examine the next argument filtering π resp. the next subset $\mathcal{D}_\mathcal{P}$.
   2.5. Otherwise, let $\mathcal{D}'$ consist of all defined symbols in $\mathcal{U}(\mathcal{D}_\mathcal{P})$.
   Try to find a quasi-simplification ordering $\succsim'$ on $\mathcal{T}(\mathcal{C} \cup \mathcal{D}', \mathcal{V})$
   and try to extend π to an argument filtering on $\mathcal{C} \cup \mathcal{D}'$
   such that $\pi(l) \succsim' \pi(r)$ for all $l \to r \in \mathcal{U}(\mathcal{D}')$.
   2.6. If Step 2.5 fails, then go to Step 2.1 and
   examine the next argument filtering π resp. the next subset $\mathcal{D}_\mathcal{P}$.
   Otherwise, continue with the next
   strongly connected component $\mathcal{P}$ in Step 2.
3. Finish with "`Termination Proved`".

For reasons of efficiency, in our implementation we did not extend size-change graphs by nodes labeled with ϵ, cf. Def. 18. These nodes would be necessary in order to simulate dependency pairs with the combined technique. Thus, if our implementation of the combined technique fails, then it might still be useful to perform an (innermost) termination proof attempt with dependency pairs.

Moreover, to increase efficiency, in our implementation of the combined approach we regard SCCs instead of cycles of the (estimated) innermost dependency graph. Clearly, every cycle is contained in a SCC and every SCC is a cycle, but a SCC may contain several (smaller) cycles. In the original dependency pair approach, using cycles leads to a more powerful technique than using SCCs for two reasons. One reason is that one can use different reduction pairs for each cycle, whereas when working with SCCs one uses the same reduction pair for all cycles contained in the SCC. However, in the examples of [2] this is not required for the innermost termination proof of any TRS. The second and more important reason why using cycles is more powerful than using SCCs is that only one dependency pair in each cycle must be strictly decreasing, whereas the others only have to be weakly decreasing. But when using SCCs, all dependency pairs of the SCC must be strictly decreasing since the SCC may consist of many cycles. (See [7, p. 51] for an example to illustrate this problem.) The constraints that all dependency pairs in a SCC are strictly decreasing are often hard to satisfy, in particular when handling mutually recursive functions.

However, this second reason is not valid any more for the combined approach where the size-change principle is integrated into dependency pairs. The reason is that the need for only one strict decrease in each cycle is implicitly covered by the size-change analysis. Thus, in the combined approach, using SCCs instead of cycles hardly changes the power of the method, whereas efficiency is increased significantly, since there are typically far less SCCs than cycles. Moreover, when regarding SCCs, in Thm. 21 (a), one must demand that all maximal multigraphs (irrespective of their labeling) contain an edge of the form $i \xrightarrow{\succsim} i$. Therefore, now extended multigraphs with labelings $(D_1, \ldots, D_n)$ and $(D'_1, \ldots, D'_m)$ are identified if their nodes and edges are identical and if $D_1 = D'_1$ and $D_n = D'_m$, but we do no longer require $\{D_1, \ldots, D_n\} = \{D'_1, \ldots, D'_m\}$. This increases efficiency, since we obtain far less possible multigraphs. One should remark that for our implementation of the original dependency pair approach, the change in efficiency is much less dramatically when using SCCs instead of cycles,[4] whereas using cycles clearly increases the power of the original approach. Therefore, we usually use cycles for the original dependency pair approach, but SCCs for the combined technique. In a future version of our implementation of the combined technique, we want to use cycles dynamically whenever the SCC-based analysis fails.

[4] The reason is that there, we start with the largest cycles and keep in mind which dependency pairs were strictly decreasing there. Then no extra work has to be done for those subcycles which contain one of these strictly decreasing dependency pairs. This approach does not work in the combination with the size-change principle, since here the selection between strict and weak decrease is hidden in the computation of maximal multigraphs.

As in Thm. 21, we only regard a reduction pair on a subset $\mathcal{D}'$ of the defined symbols. To this end, we first choose an arbitrary subset $\mathcal{D}_\mathcal{P}$ of the defined symbols occurring in right-hand sides of dependency pairs from $\mathcal{P}$. The reason is that in size-change graphs we only have symbols from $\mathcal{P}$.[5] Afterwards, we define $\mathcal{D}'$ to consist of all defined symbols occurring in $\mathcal{U}(\mathcal{D}_\mathcal{P})$. The reason is that when orienting the usable rules, we will have to consider these function symbols as well.

Now we have to generate a suitable monotonic reduction pair with orderings from $\mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{C} \cup \mathcal{D}', \mathcal{V})$ (different from Thm. 21, we do not have to extend it to tuple symbols $\mathcal{F}^\sharp$, since we do not regard nodes labeled with ϵ). As in the dependency pair approach, we use argument filterings in combination with simplification orderings for this purpose. An argument filtering π maps terms to terms by eliminating argument positions of function symbols. Moreover, it is also possible to replace all occurrences of f-terms by their i-th argument (for a function symbol f and $1 \leq i \leq \text{arity}(f)$). When computing size-change graphs, we already fix a part of the argument filtering, viz., we determine how π operates on function symbols from $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P} \cup \mathcal{D}_{\mathcal{P},\mathcal{L}}$. Here, $\mathcal{D}_{\mathcal{P},\mathcal{L}}$ denotes the set of defined symbols occurring on left-hand sides in $\mathcal{P}$. For $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}$, the argument filtering is chosen in Step 2.1 and for $\mathcal{D}_{\mathcal{P},\mathcal{L}} \setminus \mathcal{D}_\mathcal{P}$ we do not apply any filtering when constructing the size-change graphs, so we are not allowed to apply a filter later on. But we do not yet fix π on $\mathcal{C} \setminus \mathcal{C}_\mathcal{P}$ and on $\mathcal{D}' \setminus (\mathcal{D}_\mathcal{P} \cup \mathcal{D}_{\mathcal{P},\mathcal{L}})$, since all of these symbols do not occur in size-change graphs. Moreover, at this point, we still leave the simplification ordering open. Thus, for the size-change graphs we use a reduction pair $(\succsim, \succ)$ where $s \succ t$ holds iff $t \in \mathcal{T}(\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}, \mathcal{V})$ and $\pi(s) \to^+_{Emb_\mathcal{F}} \pi(t)$. Moreover, $s \succsim t$ iff $s \succ t$ or $s = t$. For reasons of efficiency, in our implementation it is possible to restrict the argument filterings considered by determining how many symbols may be filtered at most.

After computing the size-change graphs we have to calculate the maximal multigraphs and check whether all of them have an edge of the form $i \xrightarrow{\succ} i$. As this may be an expensive operation we use a cache that stores the result of this analysis for each set of size-change graphs. This caching is useful since we often have to investigate equal sets of size-change graphs that are built by different argument filters or different sets $\mathcal{D}_\mathcal{P}$. In case of success (i.e., if all maximal multigraphs have an edge of the form $i \xrightarrow{\succ} i$), the current reduction pair is refined. To this end, π is also determined on the remaining symbols from $\mathcal{C} \setminus \mathcal{C}_\mathcal{P}$ and $\mathcal{D}' \setminus (\mathcal{D}_\mathcal{P} \cup \mathcal{D}_{\mathcal{P},\mathcal{L}})$ and $\succsim$ is refined such that $s \succsim t$ iff $\pi(s) \succsim' \pi(t)$ holds for some quasi-simplification ordering $\succsim'$. Note that the quasi-ordering $\succsim$ used for the size-change graphs is indeed a subset of the refined quasi-ordering $\succsim$, since $\pi(s) \to_{Emb_\mathcal{F}} \pi(t)$ implies $\pi(s) \succsim' \pi(t)$. The reason is that a quasi-simplification ordering is a quasi-ordering containing the embedding ordering. Moreover, after this refinement, $(\succsim, \succ)$ is still a monotonic reduction pair with orderings from

[5] Defined symbols that only occur on left-hand sides of dependency pairs do not have to be included in $\mathcal{D}'$, since according to Footnote 2, we may use orderings from $\mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{C} \cup \mathcal{D}', \mathcal{V})$, i.e., the "greater" term may come from the whole signature.

$\mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{C} \cup \mathcal{D}',\mathcal{V})$. Thus, the final reduction pair used for the whole proof consists of (the extension of) $\succ$ and the refined version of $\succsim$. Since the size-change graphs were computed with a subset of the final refined quasi-ordering $\succsim$, some edges $\xrightarrow{\succsim}$ may be missing, but this only affects the power, not the soundness of the approach.

In contrast to Thm. 21, when computing size-change graphs, we do not consider the extension of the ordering $\succ$, but instead we only allow a comparison $s \succ t$ if the term t on the right-hand side is from $\succ$'s signature $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}$ for right-hand sides. This approach is still correct, since every ordering is contained in its extension. For $\succsim$, we consider a part of its extension by allowing the comparison of equal terms outside of its signature $\mathcal{C}_\mathcal{P} \cup \mathcal{D}_\mathcal{P}$ for right-hand sides. When comparing the terms of usable rules with the refined quasi-ordering over $\mathcal{T}(\mathcal{C} \cup \mathcal{D}',\mathcal{V})$, an extension is not necessary any more, since the function symbols in $\mathcal{U}(\mathcal{D}') = \mathcal{U}(\mathcal{D}_\mathcal{P})$ are already contained in $\mathcal{C} \cup \mathcal{D}'$.

The reason for only using the embedding ordering when comparing the arguments in the size-change graphs is efficiency. More sophisticated orderings like RPOS have several parameters (i.e., status and precedence). When using RPOS for ordinary termination proofs (possibly with dependency pairs), these parameters are determined incrementally. However, to transfer this approach to the size-change principle one would have to draw conclusions from an unsuccessful size-change analysis to extend the precedence. This will be done in a future version of the implementation. Nevertheless, our experiments show that the embedding ordering is sufficient for many examples. Even if we choose the embedding order for orienting the usable rules it turns out that most examples do not need a more powerful ordering.

For the original dependency pair approach, as an alternative to brute-force search we have developed an improved method which cuts off branches of the search tree which are subsumed by previously examined argument filterings. These improvements are also used in Step 2.5 in the combined technique.

To increase the power of the dependency pair approach, techniques to modify the dependency pairs by narrowing, rewriting, and instantiation were introduced in [7]. Every infinite innermost chain of the original dependency pairs corresponds to an infinite chain of the modified pairs. These refinements can also be used for the combination of dependency pairs and the size-change principle. To this end, size-change graphs are built out of the modified dependency pairs and labeled by these modified pairs. However, narrowing, rewriting, and instantiation are only applied to dependency pairs (and size-change graphs), but not to the usable rules. Here, one still uses the original rules from the TRS under consideration. In our experiments, **NRI** indicates whether the use of narrowing, rewriting, and instantiation was permitted. In that case, we always performed so called *safe transformations* which are guaranteed to terminate. After applying these safe transformations, we tried to orient the constraints resulting from the cycle. If this orientation attempt failed, at most one narrowing and one instantiation step were done for each dependency pair and then the proof attempt was repeated with the modified dependency pairs.

In addition, we have integrated a `hybrid` variant of this algorithm. The difference to the algorithm described above is the following: If Step 2.1 returns "`No Success`", then we try to solve the constraints resulting from the original dependency pair approach using SCCs. If this succeeds, we continue with the next strongly connected component in the hybrid algorithm. Otherwise we return a final "`No Success`". In combination with narrowing, rewriting, and instantiation, the hybrid algorithm first tries to use these techniques in case of a failure in Step 2.1. If these techniques do not succeed, too, then the original dependency pair approach is used for the transformed strongly connected component.

In the following experiments, we used the original dependency pair approach and the combined approach of dependency pairs with the size-change principle in order to verify innermost termination of the 110 examples in [2].[6] More precisely, we used the following **type**s of termination techniques:

- **scp** is the combination of dependency pairs and the size-change principle as described above. However, to increase efficiency, we only try sets $\mathcal{D}_{\mathcal{P}}$ with $|\mathcal{D}_{\mathcal{P}}| \leq 2$ and only allow a filtering of at most two function symbols in Step 2.1 (i.e., when building size-change graphs). Later, when orienting the usable rules in Step 2.5, one can define π on $\mathcal{C} \setminus \mathcal{C}_{\mathcal{P}}$ and $\mathcal{D}' \setminus (\mathcal{D}_{\mathcal{P}} \cup \mathcal{D}_{\mathcal{P},\mathcal{L}})$ arbitrarily.
- **hscp** is the hybrid version of **scp**.
- **dpscc** is the original dependency pair approach using SCCs instead of cycles.
- **dp** is the original dependency pair approach using cycles. However, as explained in Footnote 4, we do not check every cycle, but only the necessary ones.

In the experiments, the following base **order**s (or reduction pairs) are applied.

- **emb** is the embedding ordering. This is the weakest, but also the fastest base ordering in our experiments.
- **lpo** is the lexicographic path ordering where argument are compared lexicographically from left to right. The required precedence is determined automatically.
- **qlpo** is an extension of **lpo**. In contrast to **lpo**, in **qlpo** different symbols may be equal in the precedence (thus, "**q**" stands for "<u>q</u>uasi").
- **qrpos** is the recursive path order with status. Here, the status (i.e., multiset or lexicographic comparison w.r.t. an arbitrary permutation of the arguments) as well as the precedence is determined automatically. It subsumes all of the above orderings.

For all of these path orderings, our system offers two algorithms. In breadth-first search (**bfs**), one starts with computing a set of minimal stati and precedences which solve the first constraint. When examining the next constraint, this

[6] In Ex. 4.30c, the minus-rules must be chosen as in Ex. 4.30, 4.30a, and 4.30b. Otherwise, innermost termination can hardly be proved automatically using dependency pairs.

set is refined further, and so on. This kind of calculation is good for a fast failure detection.

Depth-first search (**dfs**) only looks for one status and precedence that solves the given constraints. If a failure is detected, backtracking is performed. Thus, this computation is usually faster than **bfs** in case of success, but slower in case of failure.

type	order	NRI	Power	Time
dpscc	emb bfs	yes	66 [60.0 %]	65.2 s [0.5 s]
dpscc	lpo bfs	yes	87 [79.0 %]	1331.6 s [12.1 s]
dp	lpo bfs	yes	93 [84.5 %]	1468.3 s [13.3 s]
dp	lpo dfs	yes	93 [84.5 %]	1474.0 s [13.4 s]
dp	qlpo bfs	yes	95 [86.3 %]	1564.0 s [14.2 s]
dp	qlpo dfs	yes	95 [86.3 %]	1679.8 s [15.2 s]
dp	qrpos bfs	yes	97 [88.1 %]	1760.9 s [16.0 s]
dp	qrpos dfs	yes	85 [77.2 %]	2481.5 s [22.5 s]
scp	emb bfs	yes	84 [76.3 %]	293.1 s [2.6 s]
scp	lpo bfs	yes	93 [84.5 %]	293.7 s [2.6 s]
scp	lpo dfs	yes	93 [84.5 %]	268.7 s [2.4 s]
scp	qlpo bfs	yes	94 [85.4 %]	262.6 s [2.3 s]
scp	qlpo dfs	yes	94 [85.4 %]	247.9 s [2.2 s]
scp	qrpos bfs	yes	94 [85.4 %]	272.7 s [2.4 s]
scp	qrpos dfs	yes	94 [85.4 %]	248.0 s [2.2 s]
hscp	lpo bfs	yes	100 [90.9 %]	423.4 s [3.8 s]
hscp	lpo dfs	yes	100 [90.9 %]	424.8 s [3.8 s]
hscp	qlpo bfs	yes	103 [93.6 %]	326.4 s [2.9 s]
hscp	qlpo dfs	yes	103 [93.6 %]	310.7 s [2.8 s]
hscp	qrpos bfs	yes	103 [93.6 %]	402.7 s [3.6 s]
hscp	qrpos dfs	yes	103 [93.6 %]	323.4 s [2.9 s]

Table 1. Performance of the different techniques on the examples of [2]

Table 1 shows in the "power" column the number and the percentage of the examples where the respective approach was successful within a time limit of 120 seconds. In the "time" column, it shows the time required for the 110 innermost termination proof attempts (where proof attempts were interrupted after 120 seconds) as well as the average time needed per example (in square brackets). Our experiments were performed on a Pentium IV with 2 GHz and 512 MB.

It turned out that RPOS had no advantage compared to LPO in our examples using the combined method, whereas RPOS was needed for four examples using the DP-approach. The possibility to regard precedences where different function symbols are equal increases the power in both methods. To analyze the specific advantages and disadvantages of dependency pairs and the combined technique, the following table presents the results when using **qrpos** as the underlying base ordering in the DP-approach, **qlpo** as the base ordering in the combined (hybrid) algorithm and when enabling the use of narrowing, rewriting, and instantiation. Thus, in this way we compare the most powerful version of dependency pairs against the most powerful version of the combined technique in our implementation. If the proof attempt finished within 120 seconds, we gave the execution time (in seconds) and otherwise we wrote "∞". Moreover, "OK" means that innermost termination was proved and "-" means that the proof attempt failed.

Type Order NRI	dp qrpos bfs yes	scp qlpo dfs yes	hscp qlpo dfs yes
#3.1	1.0 OK	0.7 OK	0.7 OK
#3.2	0.3 OK	0.1 OK	0.1 OK
#3.3	19.2 OK	0.2 OK	0.6 OK
#3.4	1.1 OK	0.2 OK	0.2 OK
#3.5	14.6 OK	0.3 OK	0.3 OK
#3.5a	10.9 OK	0.2 OK	0.2 OK
#3.5b	92.7 OK	0.3 OK	0.3 OK
#3.6	30.2 OK	0.4 OK	0.4 OK
#3.6a	15.8 OK	0.4 OK	0.4 OK
#3.6b	∞ [-]	0.5 OK	0.5 OK
#3.7	0.1 OK	0.1 OK	0.1 OK
#3.8	1.2 OK	0.3 OK	0.3 OK
#3.8a	1.3 OK	0.3 OK	0.3 OK
#3.8b	83.6 OK	0.5 OK	0.5 OK
#3.9	63.8 OK	0.3 OK	0.3 OK
#3.10	∞ [-]	69.6 -	∞ [-]
#3.11	∞ [-]	1.6 OK	2.0 OK
#3.12	3.7 -	0.9 -	1.0 -
#3.13	∞ [-]	∞ [-]	∞ [-]
#3.14	7.0 OK	0.7 -	0.8 OK
#3.15	0.3 -	0.0 -	0.1 -
#3.16	0.1 OK	0.0 OK	0.1 OK
#3.17	7.3 OK	2.0 -	2.9 OK
#3.17a	14.8 OK	2.0 -	4.3 OK
#3.18	0.4 OK	0.3 OK	0.3 OK
#3.19	0.6 OK	0.3 OK	0.4 OK
#3.20	0.6 OK	0.2 OK	0.2 OK
#3.21	0.1 OK	1.0 OK	0.5 OK
#3.22	0.5 OK	0.1 OK	0.1 OK
#3.23	0.2 OK	0.0 OK	0.1 OK
#3.24	0.5 -	0.2 -	0.3 -
#3.25	0.1 OK	0.0 OK	0.1 OK
#3.26	0.0 OK	0.2 -	0.3 OK
#3.27	0.0 OK	0.0 OK	0.0 OK
#3.28	13.5 OK	0.0 OK	0.1 OK
#3.29	0.0 OK	0.0 OK	0.0 OK
#3.30	0.0 OK	0.0 OK	0.0 OK

Type Order NRI	dp qrpos bfs yes	scp qlpo dfs yes	hscp qlpo dfs yes
#3.31	0.0 OK	0.0 OK	0.0 OK
#3.32	0.0 OK	0.0 OK	0.0 OK
#3.33	0.0 OK	0.0 OK	0.0 OK
#3.34	0.0 OK	0.0 OK	0.0 OK
#3.35	0.0 OK	0.0 OK	0.0 OK
#3.36	1.0 OK	2.0 -	2.1 OK
#3.37	0.1 OK	0.0 OK	0.0 OK
#3.38	33.7 OK	0.3 -	2.4 OK
#3.39	0.3 OK	0.2 OK	0.1 OK
#3.40	0.3 OK	0.2 OK	0.2 OK
#3.41	0.0 OK	0.0 OK	0.0 OK
#3.42	0.3 OK	0.2 OK	0.3 OK
#3.43	0.2 OK	0.1 OK	0.1 OK
#3.44	0.1 OK	0.1 OK	0.1 OK
#3.45	0.2 OK	0.3 OK	0.3 OK
#3.46	0.0 OK	0.0 OK	0.0 OK
#3.47	0.3 OK	0.1 OK	0.0 OK
#3.48	8.5 OK	6.6 -	10.2 OK
#3.49	4.1 -	0.1 -	0.6 -
#3.50	0.0 OK	0.0 OK	0.0 OK
#3.51	0.1 OK	0.0 OK	0.0 OK
#3.52	0.4 OK	0.0 OK	0.0 OK
#3.53	10.5 -	0.9 -	1.7 -
#3.53a	0.0 OK	0.0 OK	0.0 OK
#3.53b	0.4 OK	0.0 OK	0.0 OK
#3.54	0.1 OK	0.0 OK	0.1 OK
#3.55	∞ [-]	1.7 OK	1.6 OK
#3.56	0.2 OK	0.1 OK	0.1 OK
#3.57	8.2 OK	2.1 -	2.5 OK
#4.1	0.0 OK	0.0 OK	0.0 OK
#4.2	0.0 OK	0.0 OK	0.0 OK
#4.3	0.0 OK	0.0 OK	0.0 OK
#4.4	0.0 OK	0.0 OK	0.0 OK
#4.4a	0.0 OK	0.1 OK	0.1 OK
#4.5	0.0 OK	0.0 OK	0.0 OK
#4.6	0.4 OK	0.2 OK	0.2 OK
#4.7	0.0 OK	0.0 OK	0.0 OK

Type	dp		scp		hscp	
Order	qrpos	bfs	qlpo	dfs	qlpo	dfs
NRI	yes		yes		yes	
#4.8	0.7	OK	0.2	OK	0.2	OK
#4.9	1.0	OK	0.3	OK	0.3	OK
#4.10	0.0	OK	0.0	OK	0.0	OK
#4.11	0.0	OK	0.0	OK	0.0	OK
#4.12	0.8	OK	0.0	OK	0.0	OK
#4.12a	0.5	OK	0.0	OK	0.0	OK
#4.13	0.0	OK	0.0	OK	0.0	OK
#4.14	0.0	OK	0.0	OK	0.0	OK
#4.15	0.0	OK	0.0	OK	0.0	OK
#4.16	0.0	OK	0.0	OK	0.0	OK
#4.17	0.0	OK	0.0	OK	0.0	OK
#4.18	0.0	OK	0.1	OK	0.1	OK
#4.19	1.0	OK	0.0	OK	0.0	OK
#4.20	0.0	OK	0.0	OK	0.0	OK
#4.20a	0.2	OK	0.2	OK	0.1	OK
#4.21	0.1	OK	0.1	OK	0.0	OK
#4.22	0.1	OK	0.0	OK	0.0	OK
#4.23	0.4	OK	0.3	OK	0.3	OK
#4.24	0.1	OK	0.1	OK	0.0	OK

Type	dp		scp		hscp	
Order	qrpos	bfs	qlpo	dfs	qlpo	dfs
NRI	yes		yes		yes	
#4.25	0.0	OK	0.0	OK	0.0	OK
#4.26	72.0	OK	1.0	OK	1.0	OK
#4.27	0.1	OK	0.1	OK	0.1	OK
#4.28	0.2	OK	0.2	OK	0.2	OK
#4.29	111.6	OK	3.5	OK	3.6	OK
#4.30	97.5	OK	1.9	OK	1.9	OK
#4.30a	0.3	OK	0.1	OK	0.2	OK
#4.30b	45.5	OK	0.9	OK	0.9	OK
#4.30c	∞	[-]	4.6	OK	4.6	OK
#4.31	0.7	OK	0.8	OK	0.8	OK
#4.32	0.4	OK	0.0	OK	0.0	OK
#4.33	3.1	OK	0.2	OK	0.2	OK
#4.34	5.3	OK	1.1	-	2.2	OK
#4.35	∞	[-]	3.7	OK	3.7	OK
#4.36	∞	[-]	4.3	OK	4.3	OK
#4.37	0.2	OK	0.1	OK	0.1	OK
#4.37a	0.2	OK	0.1	OK	0.1	OK
Sum:	1760	97	247	94	310	103
Avg/%:	16.0	88.1	2.2	85.4	2.8	93.6

Of course, the most interesting examples are the ones where the two techniques differ in their success or in their performance.

– For some examples, dependency pairs are successful, whereas the combined non-hybrid technique fails (natural algorithms like 3.14 (comparison of binary trees), 3.17 and 3.17a (summing up list elements), 3.38 (reverse), 3.57 (comparison of binary trees + quot) and pathological examples like 3.26, 3.36, 3.48, and 4.34). In almost all of these examples, the reason is that in our implementation we only use the embedding ordering to build the size-change graphs, whereas a more sophisticated path ordering would be required here. All of these examples are easily solved with the hybrid algorithm. In the examples 3.14, 3.17, 3.17a, 3.38, 3.57, and 4.34 the hybrid algorithm is even faster than the dependency pair technique. The reason is that the combined technique can determine quickly that it fails without narrowing, rewriting, or instantiation, and so the transformations will be applied earlier. After the transformation limit has been reached, the system switches to the original dependency pair approach and the transformed pairs can be oriented directly. If one starts with the dependency pair method, the need for transformations is determined only after a failed orientation attempt with the use of dependency pairs. The detection of this failure costs more time than when using the combined algorithm.

– For some examples, the combined technique is successful, whereas dependency pairs "fail" (3.6b (gcd), 3.11 (quicksort), 3.55 (quicksort + div), 4.30c (gcd), 4.35 (renaming in lambda calculus), 4.36 (selection sort)) or take much longer time (3.5b (mod), 3.6 (gcd), 3.8b (log), 4.26 (minus), 4.29 (times), 4.30 (quot), 4.30b (mod)). In all the "failures" mentioned above, the proof attempt was aborted because of a timeout. Indeed, for all these example it is possible to solve them with the dependency pair approach if one would only allow them enough time. One might argue that this comparison is unfair because in the combined method we used the faster **qlpo**. But for examples 3.6, 3.6a, 3.6b, and 4.30c one really needs the slower but more powerful **qrpos**. (In contrast, for three of these four examples the combined algorithm is successful even with the embedding order.) The time difference between the combined approach and the original dependency pair technique can even be seen if one uses **qrpos** in the combined approach (for the examples mentioned above, the combined approach using **qrpos** is at most 5 seconds slower; see the full tables at the end of this report for details). In the other examples, the combined technique benefits from mainly two facts. There are less usable rules, and there are far less argument filterings that have to be considered for orienting the usable rules than in the original approach: If we perform a successful size-change analysis, we have fixed the argument filtering for many symbols, so the search space is reduced enormously.
– Finally, there are 7 examples which cannot be handled by our current implementation (3.10 (minsort), 3.12 (shuffle), 3.13 (reachability), 3.15 (average), 3.53 (quot + shuffle + comparison of binary trees), as well as the two pathological examples 3.24 and 3.49).
  Termination of Example 3.24 cannot be shown using the combined approach or the dependency pair method if one is restricted to path orderings like RPOS. But if we use a reduction pair based on Knuth-Bendix orderings or on polynomial orderings then termination is easy to prove.
  In the remaining examples 3.10, 3.12, 3.13, 3.15, 3.49, and 3.53, path orderings like RPOS are too weak for a successful termination proof, too. The difference to Example 3.24 is that here even Knuth-Bendix orderings cannot be used to generate appropriate reduction pairs. But again, with reduction pairs based on polynomial orderings our algorithm would be able to prove the termination of these examples. We are currently working on an implementation of polynomial orderings in AProVE.

To summarize, we implemented a version of the combined technique which uses SCCs instead of cycles, which disregards nodes of size-change graphs labeled with ϵ, and which only builds size-change graphs using the embedding ordering. The advantage is that in this way, the method works very efficiently and our experiments show that this approach is already very powerful. As a consequence of the efficiency of the basic algorithm, we have developed a hybrid variant where we first do a fast and often successful analysis based on the combined technique and in case of a failure we switch to the original dependency pair method. In this way we solved 103 of the 110 examples with **qlpo** as underlying reduction pair. For each of these examples, the proof is performed in less than 10.5 seconds (and most examples are solved in less than a second).

The detailed results of our experiments can be found on the following pages. All experiments in the following tables were performed completely automatically.

Type Order NRI	dpscc emb bfs yes	dpscc lpo bfs yes	dp lpo bfs yes	dp lpo dfs yes	dp qlpo bfs yes	dp qlpo dfs yes	dp qrpos bfs yes	dp qrpos dfs yes
#3.1	0.5 OK	0.5 OK	0.6 OK	0.6 OK	0.6 OK	0.6 OK	1.0 OK	0.8 OK
#3.2	0.1 OK	0.1 OK	0.1 OK	0.7 OK	0.1 OK	0.1 OK	0.3 OK	0.1 OK
#3.3	0.2 -	6.2 OK	6.3 OK	6.4 OK	9.3 OK	9.2 OK	19.2 OK	∞ [-]
#3.4	0.1 -	0.2 OK	0.2 OK	0.2 OK	0.3 OK	0.2 OK	1.1 OK	0.3 OK
#3.5	0.3 -	6.8 OK	7.1 OK	7.3 OK	7.7 OK	10.6 OK	14.6 OK	∞ [-]
#3.5a	0.3 -	8.3 OK	5.8 OK	6.0 OK	6.3 OK	9.5 OK	10.9 OK	∞ [-]
#3.5b	0.3 -	33.5 OK	33.3 OK	33.4 OK	70.4 OK	98.2 OK	92.7 OK	∞ [-]
#3.6	0.4 -	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	30.2 OK	∞ [-]
#3.6a	0.4 -	68.4 -	∞ [-]	∞ [-]	∞ [-]	∞ [-]	15.8 OK	∞ [-]
#3.6b	0.3 -	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]
#3.7	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.1 OK	0.1 OK
#3.8	0.1 OK	0.4 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	1.2 OK	0.5 OK
#3.8a	0.3 OK	0.3 OK	0.4 OK	0.3 OK	0.4 OK	0.4 OK	1.3 OK	0.6 OK
#3.8b	0.3 -	28.5 OK	27.0 OK	27.1 OK	44.1 OK	31.9 OK	83.6 OK	∞ [-]
#3.9	0.4 -	28.8 OK	29.5 OK	29.4 OK	38.5 OK	44.8 OK	63.8 OK	∞ [-]
#3.10	1.3 -	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]
#3.11	0.7 -	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]
#3.12	0.2 -	0.4 -	0.4 -	0.4 -	0.5 -	0.5 -	3.7 -	3.0 -
#3.13	3.2 -	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]
#3.14	0.2 -	0.3 -	0.8 -	0.8 -	0.4 OK	0.3 OK	7.0 OK	1.6 OK
#3.15	0.0 -	0.0 -	0.0 -	0.1 -	0.0 -	0.1 -	0.3 -	0.3 -
#3.16	0.0 OK	0.1 OK	0.1 OK	0.1 OK	0.0 OK	0.1 OK	0.1 OK	0.1 OK
#3.17	0.4 -	0.9 OK	1.0 OK	1.0 OK	1.0 OK	1.0 OK	7.3 OK	2.5 OK
#3.17a	0.4 -	3.1 OK	2.9 OK	3.0 OK	3.5 OK	3.4 OK	14.8 OK	9.0 OK
#3.18	0.5 -	0.1 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.4 OK	0.2 OK
#3.19	0.6 -	0.2 OK	0.3 OK	0.2 OK	0.3 OK	0.2 OK	0.6 OK	0.3 OK
#3.20	0.1 OK	0.2 OK	0.5 OK	0.2 OK	0.1 OK	0.1 OK	0.6 OK	0.2 OK
#3.21	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.22	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.3 OK	0.5 OK	0.4 OK
#3.23	0.0 OK	0.1 OK	0.1 OK	0.0 OK	0.5 OK	0.1 OK	0.2 OK	0.1 OK
#3.24	0.1 -	0.2 -	0.3 -	0.2 -	0.3 -	0.3 -	0.5 -	0.5 -
#3.25	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK
#3.26	0.1 -	0.4 -	0.2 -	0.2 -	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.27	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.28	0.5 -	0.8 OK	0.8 OK	0.8 OK	1.3 OK	0.9 OK	13.5 OK	3.1 OK
#3.29	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.30	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.31	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.32	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.33	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.34	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.35	0.0 OK	0.1 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.0 OK	0.0 OK
#3.36	0.9 -	0.7 OK	1.6 OK	1.5 OK	0.6 OK	0.5 OK	1.0 OK	0.7 OK

Type	dpscc	dpscc	dp	dp	dp	dp	dp	dp
Order	emb bfs	lpo bfs	lpo bfs	lpo dfs	qlpo bfs	qlpo dfs	qrpos bfs	qrpos dfs
NRI	yes	yes	yes	yes	yes	yes	yes	yes
#3.37	0.0 -	0.0 -	0.0 OK	0.1 OK	0.0 OK	0.0 OK	0.1 OK	0.1 OK
#3.38	0.2 -	1.8 OK	2.2 OK	2.2 OK	3.9 OK	5.8 OK	33.7 OK	29.1 OK
#3.39	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.3 OK	0.2 OK
#3.40	0.1 OK	0.2 OK	0.3 OK	0.2 OK	0.2 OK	0.2 OK	0.3 OK	0.2 OK
#3.41	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.42	0.1 OK	0.1 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.3 OK	0.2 OK
#3.43	0.0 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.2 OK	0.2 OK
#3.44	0.0 OK	0.1 OK	0.0 OK	0.0 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.45	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.2 OK	0.2 OK	0.2 OK	0.1 OK
#3.46	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.47	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.1 OK	0.0 OK	0.3 OK	0.1 OK
#3.48	0.5 -	3.8 OK	3.9 OK	4.0 OK	4.3 OK	3.8 OK	8.5 OK	4.7 OK
#3.49	0.0 -	0.0 -	2.2 -	2.0 -	2.1 -	2.0 -	4.1 -	4.0 -
#3.50	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.51	0.0 OK	0.1 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.1 OK	0.0 OK
#3.52	0.0 -	0.2 -	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.4 OK	0.2 OK
#3.53	0.5 -	0.6 -	1.1 -	1.3 -	1.2 -	1.0 -	10.5 -	4.5 -
#3.53a	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.53b	0.0 -	0.1 -	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.4 OK	0.2 OK
#3.54	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.1 OK	0.0 OK	0.1 OK	0.1 OK
#3.55	0.7 -	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]
#3.56	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.1 OK	0.1 OK	0.2 OK	0.1 OK
#3.57	0.3 -	1.2 OK	1.1 OK	1.1 OK	1.3 OK	1.2 OK	8.2 OK	3.1 OK
#4.1	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.2	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.3	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.4	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.4a	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.5	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.6	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.4 OK	0.1 OK
#4.7	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.8	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.7 OK	0.4 OK
#4.9	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.2 OK	0.2 OK	1.0 OK	0.6 OK
#4.10	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.11	0.1 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.12	0.0 OK	0.1 OK	0.2 OK	0.2 OK	0.3 OK	0.4 OK	0.8 OK	0.6 OK
#4.12a	0.0 OK	0.1 OK	0.2 OK	0.2 OK	0.2 OK	0.1 OK	0.5 OK	0.4 OK
#4.13	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.14	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.15	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.16	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.17	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.18	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK
#4.19	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	1.0 OK	0.2 OK
#4.20	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.20a	0.0 OK	0.0 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.2 OK	0.1 OK
#4.21	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.1 OK	0.0 OK
#4.22	0.0 -	0.0 -	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.1 OK

Type	dpscc		dpscc		dp		dp		dp		dp		dp		dp	
Order	emb	bfs	lpo	bfs	lpo	bfs	lpo	dfs	qlpo	bfs	qlpo	dfs	qrpos	bfs	qrpos	dfs
NRI	yes		yes		yes		yes		yes		yes		yes		yes	
#4.23	0.1	-	0.2	-	0.1	OK	0.1	OK	0.2	OK	0.1	OK	0.4	OK	0.2	OK
#4.24	0.0	-	0.1	-	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#4.25	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.26	0.2	-	10.3	OK	23.3	OK	24.8	OK	29.0	OK	46.2	OK	72.0	OK	∞	[-]
#4.27	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#4.28	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.2	OK	0.2	OK
#4.29	0.5	-	16.1	OK	42.8	OK	43.2	OK	53.2	OK	95.6	OK	111.6	OK	∞	[-]
#4.30	0.3	-	23.0	OK	41.8	OK	44.2	OK	48.7	OK	68.8	OK	97.5	OK	∞	[-]
#4.30a	0.1	OK	0.1	OK	0.4	OK	0.1	OK	0.1	OK	0.1	OK	0.3	OK	0.1	OK
#4.30b	0.3	-	13.3	OK	20.6	OK	22.0	OK	23.3	OK	32.2	OK	45.5	OK	∞	[-]
#4.30c	0.6	-	103.2	-	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]
#4.31	0.2	OK	0.5	OK	0.4	OK	0.4	OK	0.4	OK	0.5	OK	0.7	OK	0.6	OK
#4.32	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.1	OK	0.0	OK	0.4	OK	0.1	OK
#4.33	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.4	OK	0.3	OK	3.1	OK	0.5	OK
#4.34	0.4	-	1.2	OK	1.2	OK	1.1	OK	1.4	OK	1.2	OK	5.3	OK	1.5	OK
#4.35	39.7	-	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]
#4.36	0.5	-	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]
#4.37	0.1	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.1	OK	0.2	OK	0.1	OK
#4.37a	0.0	OK	0.0	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.2	OK	0.1	OK
Sum:	65	66	1331	87	1468	93	1474	93	1564	95	1679	95	1760	97	2481	85
Avg/%:	0.5	60.0	12.1	79.0	13.3	84.5	13.4	84.5	14.2	86.3	15.2	86.3	16.0	88.1	22.5	77.2

Type	scp	scp	scp	scp	scp	scp	scp
Order	emb bfs	lpo bfs	lpo dfs	qlpo bfs	qlpo dfs	qrpos bfs	qrpos dfs
NRI	yes	yes	yes	yes	yes	yes	yes
#3.1	0.6 OK	0.7 OK	0.6 OK	0.6 OK	0.7 OK	0.7 OK	0.7 OK
#3.2	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.3	1.0 -	0.2 OK	0.2 OK	0.7 OK	0.2 OK	0.4 OK	0.2 OK
#3.4	0.8 -	0.2 OK	0.2 OK	0.1 OK	0.2 OK	0.7 OK	0.2 OK
#3.5	0.3 OK	0.3 OK	0.3 OK	1.0 OK	0.3 OK	0.3 OK	0.3 OK
#3.5a	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK
#3.5b	2.2 -	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK
#3.6	0.4 OK	0.4 OK	0.5 OK	0.4 OK	0.4 OK	0.4 OK	0.4 OK
#3.6a	0.4 OK	0.6 OK	0.4 OK	0.5 OK	0.4 OK	0.4 OK	0.4 OK
#3.6b	2.9 -	0.5 OK	0.7 OK	0.5 OK	0.5 OK	0.5 OK	0.5 OK
#3.7	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.8	0.3 OK	0.2 OK	0.3 OK	0.2 OK	0.3 OK	0.2 OK	0.2 OK
#3.8a	0.3 OK	0.2 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK
#3.8b	1.4 -	0.5 OK	0.5 OK	0.5 OK	0.5 OK	0.7 OK	0.7 OK
#3.9	2.0 -	0.3 OK	0.3 OK	0.4 OK	0.3 OK	0.6 OK	0.4 OK
#3.10	69.5 -	68.1 -	68.3 -	69.0 -	69.6 -	68.7 -	68.7 -
#3.11	6.1 -	1.6 OK	1.6 OK	1.7 OK	1.6 OK	2.4 OK	1.7 OK
#3.12	0.8 -	0.8 -	0.8 -	0.8 -	0.9 -	1.0 -	1.0 -
#3.13	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]	∞ [-]
#3.14	0.7 -	0.7 -	0.6 -	0.6 -	0.7 -	0.7 -	0.6 -
#3.15	0.0 -	0.0 -	0.0 -	0.0 -	0.0 -	0.0 -	0.0 -
#3.16	0.0 OK	0.1 OK	0.1 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK
#3.17	2.2 -	2.2 -	1.9 -	1.9 -	2.0 -	2.3 -	1.9 -
#3.17a	1.8 -	1.8 -	2.0 -	2.0 -	2.0 -	1.8 -	2.0 -
#3.18	0.2 OK	0.2 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK
#3.19	0.3 OK	0.3 OK	0.4 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK
#3.20	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK
#3.21	0.4 OK	0.4 OK	0.9 OK	0.4 OK	1.0 OK	0.4 OK	0.4 OK
#3.22	0.1 OK	0.1 OK	0.0 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.23	0.1 OK	0.1 OK	0.0 OK	0.1 OK	0.0 OK	0.1 OK	0.1 OK
#3.24	0.2 -	0.2 -	0.2 -	0.2 -	0.2 -	0.2 -	0.2 -
#3.25	0.0 OK	0.1 OK	0.1 OK	0.1 OK	0.0 OK	0.1 OK	0.1 OK
#3.26	0.2 -	0.2 -	0.2 -	0.2 -	0.2 -	0.2 -	0.2 -
#3.27	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.5 OK
#3.28	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.0 OK	0.1 OK	0.0 OK
#3.29	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.30	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.31	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.32	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.33	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.34	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.35	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.36	2.1 -	2.2 -	2.0 -	1.9 -	2.0 -	2.2 -	1.9 -
#3.37	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.38	0.3 -	0.3 -	0.3 -	0.3 -	0.3 -	0.3 -	0.3 -
#3.39	0.5 OK	0.2 OK	0.2 OK	0.1 OK	0.2 OK	0.2 OK	0.2 OK
#3.40	0.2 OK	0.6 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK
#3.41	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK

Type	scp	scp	scp	scp	scp	scp	scp
Order	emb bfs	lpo bfs	lpo dfs	qlpo bfs	qlpo dfs	qrpos bfs	qrpos dfs
NRI	yes	yes	yes	yes	yes	yes	yes
#3.42	0.2 OK	0.2 OK	0.3 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK
#3.43	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.44	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.45	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK
#3.46	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.47	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.0 OK
#3.48	5.5 -	5.5 -	6.0 -	5.7 -	6.6 -	5.5 -	5.7 -
#3.49	0.1 -	0.1 -	0.1 -	0.1 -	0.1 -	0.1 -	1.0 -
#3.50	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.51	0.1 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK
#3.52	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.53	1.0 -	1.0 -	0.9 -	0.9 -	0.9 -	1.0 -	0.9 -
#3.53a	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.53b	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#3.54	0.1 OK	0.1 OK	0.0 OK	0.1 OK	0.0 OK	0.1 OK	0.1 OK
#3.55	7.0 -	1.8 OK	1.7 OK	1.8 OK	1.7 OK	2.2 OK	1.7 OK
#3.56	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#3.57	2.2 -	4.1 -	2.1 -	2.0 -	2.1 -	4.2 -	2.1 -
#4.1	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.2	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.3	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.4	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.4a	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.0 OK
#4.5	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.6	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.9 OK	0.2 OK
#4.7	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.8	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK	0.2 OK
#4.9	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK	0.3 OK
#4.10	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.11	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.12	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK
#4.12a	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.13	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.14	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.15	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.16	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.2 OK
#4.17	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.18	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK
#4.19	0.1 OK	0.0 OK	0.1 OK	0.1 OK	0.0 OK	0.0 OK	0.1 OK
#4.20	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.20a	0.1 OK	0.1 OK	0.2 OK	0.2 OK	0.2 OK	0.1 OK	0.2 OK
#4.21	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.0 OK
#4.22	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.23	0.3 OK	0.2 OK	0.3 OK	0.3 OK	0.3 OK	0.2 OK	0.3 OK
#4.24	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.1 OK	0.0 OK	0.0 OK
#4.25	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK	0.0 OK
#4.26	0.9 OK	0.9 OK	2.9 OK	0.9 OK	1.0 OK	0.9 OK	1.0 OK
#4.27	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK	0.1 OK

Type	scp		scp		scp		scp		scp		scp		scp	
Order	emb	bfs	lpo	bfs	lpo	dfs	qlpo	bfs	qlpo	dfs	qrpos	bfs	qrpos	dfs
NRI	yes		yes		yes		yes		yes		yes		yes	
#4.28	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#4.29	1.5	OK	1.5	OK	1.4	OK	2.7	OK	3.5	OK	1.6	OK	3.8	OK
#4.30	2.0	OK	2.1	OK	1.9	OK	1.8	OK	1.9	OK	2.1	OK	1.8	OK
#4.30a	0.1	OK	0.1	OK	0.2	OK	0.1	OK	0.1	OK	0.1	OK	0.2	OK
#4.30b	0.9	OK	1.3	OK	0.9	OK	0.9	OK	0.9	OK	2.4	OK	0.9	OK
#4.30c	6.2	OK	4.1	OK	4.6	OK	4.5	OK	4.6	OK	4.2	OK	4.5	OK
#4.31	0.7	OK	0.8	OK	1.1	OK	0.8	OK	0.8	OK	0.7	OK	0.8	OK
#4.32	0.0	OK	0.1	OK	0.0	OK	0.1	OK	0.0	OK	0.0	OK	0.1	OK
#4.33	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#4.34	0.7	-	0.7	-	0.7	-	0.8	-	1.1	-	0.8	-	0.8	-
#4.35	27.7	-	52.1	-	26.9	-	4.4	OK	3.7	OK	5.6	OK	4.4	OK
#4.36	10.1	-	4.4	OK	4.3	OK	21.0	OK	4.3	OK	25.3	OK	4.5	OK
#4.37	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#4.37a	0.1	OK	0.4	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
Sum:	293	84	293	93	268	93	262	94	247	94	272	94	248	94
Avg/%:	2.6	76.3	2.6	84.5	2.4	84.5	2.3	85.4	2.2	85.4	2.4	85.4	2.2	85.4

Type	hscp		hscp		hscp		hscp		hscp		hscp	
Order	lpo	bfs	lpo	dfs	qlpo	bfs	qlpo	dfs	qrpos	bfs	qrpos	dfs
NRI	yes		yes		yes		yes		yes		yes	
#3.1	0.7	OK	0.6	OK	0.7	OK	0.7	OK	0.7	OK	0.6	OK
#3.2	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#3.3	0.2	OK	0.2	OK	0.2	OK	0.6	OK	0.3	OK	0.2	OK
#3.4	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#3.5	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK
#3.5a	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.3	OK	0.2	OK
#3.5b	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK
#3.6	0.5	OK	0.4	OK	0.4	OK	0.4	OK	0.4	OK	0.4	OK
#3.6a	0.4	OK	0.4	OK	0.4	OK	0.4	OK	0.4	OK	0.4	OK
#3.6b	0.5	OK	0.5	OK	0.5	OK	0.5	OK	0.6	OK	0.5	OK
#3.7	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#3.8	0.2	OK	0.2	OK	0.3	OK	0.3	OK	0.2	OK	0.2	OK
#3.8a	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK
#3.8b	0.5	OK	0.5	OK	0.5	OK	0.5	OK	0.8	OK	0.7	OK
#3.9	0.3	OK	0.3	OK	0.4	OK	0.3	OK	0.5	OK	0.3	OK
#3.10	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]
#3.11	1.7	OK	2.0	OK	2.1	OK	2.0	OK	2.5	OK	2.0	OK
#3.12	1.0	-	1.0	-	1.0	-	1.0	-	3.9	-	2.2	-
#3.13	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]	∞	[-]
#3.14	1.2	-	1.3	-	0.8	OK	0.8	OK	6.8	OK	1.8	OK
#3.15	0.0	-	0.1	-	0.1	-	0.1	-	0.1	-	0.1	-
#3.16	0.0	OK	0.0	OK	0.0	OK	0.1	OK	0.1	OK	0.1	OK
#3.17	2.1	OK	2.4	OK	2.4	OK	2.9	OK	7.7	OK	3.3	OK
#3.17a	4.0	OK	4.2	OK	4.4	OK	4.3	OK	14.5	OK	6.2	OK
#3.18	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK
#3.19	0.3	OK	0.4	OK	0.3	OK	0.4	OK	0.3	OK	0.3	OK
#3.20	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#3.21	0.4	OK	0.4	OK	0.4	OK	0.5	OK	0.4	OK	0.4	OK
#3.22	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#3.23	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#3.24	0.3	-	0.3	-	0.7	-	0.3	-	0.4	-	0.3	-
#3.25	0.0	OK	0.4	OK	0.0	OK	0.1	OK	0.1	OK	0.1	OK
#3.26	0.2	-	0.2	-	0.2	OK	0.3	OK	0.8	OK	0.3	OK
#3.27	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.28	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.0	OK	0.1	OK
#3.29	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.30	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.31	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.32	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.33	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.34	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.35	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.36	2.7	OK	2.0	OK	2.1	OK	2.1	OK	2.4	OK	2.8	OK
#3.37	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.38	2.1	OK	2.0	OK	2.7	OK	2.4	OK	6.3	OK	3.8	OK
#3.39	0.2	OK	0.2	OK	0.1	OK	0.1	OK	0.2	OK	0.1	OK
#3.40	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#3.41	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK

Type Order NRI	hscp lpo bfs yes		hscp lpo dfs yes		hscp qlpo bfs yes		hscp qlpo dfs yes		hscp qrpos bfs yes		hscp qrpos dfs yes	
#3.42	0.2	OK	0.2	OK	0.2	OK	0.3	OK	0.2	OK	0.2	OK
#3.43	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#3.44	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#3.45	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK
#3.46	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.47	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.48	10.0	OK	10.2	OK	10.9	OK	10.2	OK	31.2	OK	11.1	OK
#3.49	0.1	-	0.1	-	0.1	-	0.6	-	1.4	-	0.5	-
#3.50	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.51	0.1	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.52	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.53	1.0	-	0.9	-	1.9	-	1.7	-	9.7	-	4.9	-
#3.53a	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.53b	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#3.54	0.1	OK	0.0	OK	0.0	OK	0.1	OK	0.0	OK	0.0	OK
#3.55	1.8	OK	1.7	OK	1.7	OK	1.6	OK	2.0	OK	1.7	OK
#3.56	1.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#3.57	2.3	OK	2.5	OK	2.5	OK	2.5	OK	7.6	OK	2.9	OK
#4.1	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.2	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.3	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.4	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.4a	0.0	OK	0.0	OK	0.0	OK	0.1	OK	0.0	OK	0.0	OK
#4.5	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.6	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#4.7	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.8	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#4.9	0.4	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.5	OK
#4.10	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.11	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.12	0.0	OK	0.1	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.12a	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.13	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.14	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.15	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.16	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.17	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.18	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#4.19	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.1	OK	0.0	OK
#4.20	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.20a	0.1	OK	0.1	OK	0.2	OK	0.1	OK	0.2	OK	0.1	OK
#4.21	0.0	OK	0.1	OK	0.1	OK	0.0	OK	0.0	OK	0.0	OK
#4.22	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.23	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK	0.3	OK
#4.24	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.1	OK	0.0	OK
#4.25	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK
#4.26	0.9	OK	1.0	OK	1.0	OK	1.0	OK	1.0	OK	1.0	OK
#4.27	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK

Type	hscp		hscp		hscp		hscp		hscp		hscp	
Order	lpo	bfs	lpo	dfs	qlpo	bfs	qlpo	dfs	qrpos	bfs	qrpos	dfs
NRI	yes		yes		yes		yes		yes		yes	
#4.28	0.2	OK	0.2	OK	0.8	OK	0.2	OK	0.3	OK	0.2	OK
#4.29	1.6	OK	3.8	OK	1.4	OK	3.6	OK	3.3	OK	1.7	OK
#4.30	2.9	OK	1.9	OK	1.9	OK	1.9	OK	1.9	OK	4.1	OK
#4.30a	0.1	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.1	OK
#4.30b	0.8	OK	0.9	OK	0.9	OK	0.9	OK	0.9	OK	0.9	OK
#4.30c	4.3	OK	4.5	OK	4.6	OK	4.6	OK	4.6	OK	4.5	OK
#4.31	0.8	OK	0.8	OK	1.2	OK	0.8	OK	1.2	OK	0.8	OK
#4.32	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.0	OK	0.1	OK
#4.33	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK	0.2	OK
#4.34	2.0	OK	2.2	OK	2.1	OK	2.2	OK	8.0	OK	2.3	OK
#4.35	∞	[-]	∞	[-]	3.9	OK	3.7	OK	5.2	OK	4.7	OK
#4.36	4.3	OK	4.2	OK	20.9	OK	4.3	OK	24.0	OK	4.5	OK
#4.37	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
#4.37a	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK	0.1	OK
Sum:	423	100	424	100	326	103	310	103	402	103	323	103
Avg/%:	3.8	90.9	3.8	90.9	2.9	93.6	2.8	93.6	3.6	93.6	2.9	93.6

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

95-11 *	M. Staudt / K. von Thadden: Subsumption Checking in Knowledge Bases
95-12 *	G.V. Zemanek / H.W. Nissen / H. Hubert / M. Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
95-13 *	M. Staudt / M. Jarke: Incremental Maintenance of Externally Materialized Views
95-14 *	P. Peters / P. Szczurko / M. Jeusfeld: Business Process Oriented Information Management: Conceptual Models at Work
95-15 *	S. Rams / M. Jarke: Proceedings of the Fifth Annual Workshop on Information Technologies & Systems
95-16 *	W. Hans / St. Winkler / F. Sáenz: Distributed Execution in Functional Logic Programming
96-1 *	Jahresbericht 1995
96-2	M. Hanus / Chr. Prehofer: Higher-Order Narrowing with Definitional Trees
96-3 *	W. Scheufele / G. Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
96-4	K. Pohl: PRO-ART: Enabling Requirements Pre-Traceability
96-5	K. Pohl: Requirements Engineering: An Overview
96-6 *	M. Jarke / W. Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
96-7	O. Chitil: The ς-Semantics: A Comprehensive Semantics for Functional Programs
96-8 *	S. Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
96-9	M. Hanus (Ed.): Proceedings of the Poster Session of ALP'96 — Fifth International Conference on Algebraic and Logic Programming
96-10	R. Conradi / B. Westfechtel: Version Models for Software Configuration Management
96-11 *	C. Weise / D. Lenzkes: A Fast Decision Algorithm for Timed Refinement
96-12 *	R. Dömges / K. Pohl / M. Jarke / B. Lohmann / W. Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
96-13 *	K. Pohl / R. Klamma / K. Weidenhaupt / R. Dömges / P. Haumer / M. Jarke: A Framework for Process-Integrated Tools

96-14 * R. Gallersdörfer / K. Klabunde / A. Stolz / M. Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996

96-15 * H. Schimpe / M. Staudt: VAREX: An Environment for Validating and Refining Rule Bases

96-16 * M. Jarke / M. Gebhardt, S. Jacobs, H. Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

96-17 M. Jeusfeld / T. X. Bui: Decision Support Components on the Internet

96-18 M. Jeusfeld / M. Papazoglou: Information Brokering: Design, Search and Transformation

96-19 * P. Peters / M. Jarke: Simulating the impact of information flows in networked organizations

96-20 M. Jarke / P. Peters / M. Jeusfeld: Model-driven planning and design of cooperative information systems

96-21 * G. de Michelis / E. Dubois / M. Jarke / F. Matthes / J. Mylopoulos / K. Pohl / J. Schmidt / C. Woo / E. Yu: Cooperative information systems: a manifesto

96-22 * S. Jacobs / M. Gebhardt, S. Kethers, W. Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

96-23 * M. Gebhardt / S. Jacobs: Conflict Management in Design

97-01 Jahresbericht 1996

97-02 J. Faassen: Using full parallel Boltzmann Machines for Optimization

97-03 A. Winter / A. Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

97-04 M. Mohnen / S. Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

97-05 * S. Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

97-06 M. Nicola / M. Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

97-07 P. Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

97-08 D. Blostein / A. Schürr: Computing with Graphs and Graph Rewriting

97-09 C.-A. Krapp / B. Westfechtel: Feedback Handling in Dynamic Task Nets

97-10 M. Nicola / M. Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

97-13 M. Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

97-14 R. Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

97-15 G. H. Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

98-01 * Jahresbericht 1997

98-02 S. Gruner/ M. Nagel / A. Schürr: Fine-grained and Structure-oriented Integration Tools are Needed for Product Development Processes

98-03 S. Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

98-04 * O. Kubitz: Mobile Robots in Dynamic Environments

98-05 M. Leucker / St. Tobies: Truth — A Verification Platform for Distributed Systems

98-07 M. Arnold / M. Erdmann / M. Glinz / P. Haumer / R. Knoll / B. Paech / K. Pohl / J. Ryser / R. Studer / K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

98-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen

98-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

98-10 * M. Nicola / M. Jarke: Performance Modeling of Distributed and Replicated Databases

98-11 * A. Schleicher / B. Westfechtel / D. Jäger: Modeling Dynamic Software Processes in UML

98-12 * W. Appelt / M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

98-13 K. Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

99-01 * Jahresbericht 1998

99-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

99-03 * R. Gallersdörfer / M. Jarke / M. Nicola: The ADR Replication Manager

99-04 M. Alpuente / M. Hanus / S. Lucas / G. Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

99-07 Th. Wilke: CTL+ is exponentially more succinct than CTL

99-08 O. Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge / Marcin Jurdzinski: A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-04 Andreas Becks / Stefan Sklorz / Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 Mareike Schoop: Cooperative Document Management

2000-06 Mareike Schoop / Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen / Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08 Thomas Arts / Thomas Noll: Verifying Generic Erlang Client-Server Implementations

2001-01 * Jahresbericht 2000

2001-02 Benedikt Bollig / Martin Leucker: Deciding LTL over Mazurkiewicz Traces

2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04 Benedikt Bollig / Martin Leucker / Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus

2001-05 Benedikt Bollig / Martin Leucker / Thomas Noll: Regular MSC Languages

2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic

2001-07 Martin Grohe / Stefan Wöhrle: An Existential Locality Theorem

2001-08 Mareike Schoop / James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling

2001-09 Thomas Arts / Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs

2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures

2001-11 Klaus Indermark / Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung

2002-01 * Jahresbericht 2001

2002-02 Jürgen Giesl / Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems

2002-03 Benedikt Bollig / Martin Leucker / Thomas Noll: Generalised Regular MSC Languages

2002-04 Jürgen Giesl / Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting

2002-05 Horst Lichter / Thomas von der Maßen / Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines

2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata

2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities

2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java

2002-09 Markus Mohnen: Interfaces with Default Implementations in Java

2002-10 Martin Leucker: Logics for Mazurkiewicz traces

2002-11 Jürgen Giesl / Hans Zantema: Liveness in Rewriting

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.